Controllo remoto ad IR per il Boe-Bot

Scritto da Andy Lindsay

VERSIONE 1.0

PARALLAX INC

Garanzia

La Parallax garantisce i suoi prodotti contro difetti dei materiali e di manifattura per un periodo di 90 giorni dal ricevimento del prodotto. Se scoprite un difetto, Parallax, a suo giudizio, lo riparerà o sostituirà la merce o rimborserà il prezzo di acquisto. Prima di restituire il prodotto a Parallax, richiedete un numero Return Merchandise Authorization (RMA). Scrivere il numero RMA fuori dalla scatola utilizzata per restituire la merce a Parallax. Per favore includete le seguenti informazioni insieme alla merce restituita: il vostro nome, numero telefonico, indirizzo della spedizione e una descrizione del problema. La Parallax restituirà il vostro prodotto o la sua sostituzione utilizzando lo stesso metodo di spedizione utilizzato per inviare il prodotto a Parallax.

Garanzia soddisfatti o rimborsati di 14 giorni

Se, entro 14 giorni dal ricevimento del vostro prodotto, trovate che non soddisfa le vostre necessità, potete restituirlo per un rimborso completo. La Parallax rimborserà il prezzo di acquisto del prodotto, esclusi costi di spedizione/gestione. Questa garanzia è nulla se il prodotto è stato modificato o danneggiato.

Copyright e marchi

ISBN 1-928982-31-X

Rinuncia di responsabilità

La Parallax, Inc. non è responsabile di danni, speciali, accidentali o conseguenti che derivano da qualsiasi violazione della garanzia o sotto qualsiasi teoria legale, includendo utili persi, tempo di fermo, buonuscita, danno a o sostituzione dell'apparecchiatura o di proprietà o qualsiasi costo di ripristino, riprogrammazione o riproduzione di qualsiasi dato memorizzato in o utilizzato con in prodotti Parallax. La Parallax non è inoltre responsabile di qualsiasi danno personale, tra cui quello alla vita e alla salute, che derivano da utilizzo di un qualunque dei nostri prodotti. Vi assumete la responsabilità completa per la vostra applicazione BASIC Stamp, qualsiasi rischio per la vita possa costituire.

Sito WEB e Liste di Discussione

Il sito web www.parallax.com possiede molti canali per il download, prodotti, applicativi per i clienti, ed acquisti on-line per i componenti usati in questo testo. Manteniamo inoltre diverse liste di discussione e-mail per persone interessate nell'uso dei prodotti Parallax. Queste liste sono accessibili dal sito www.parallax.com attraverso il menù Support ⟶ Discussion Groups. Le liste con cui noi operiamo sono le seguenti:

- BASIC Stamps – Con oltre 2,500 sottoscrittori, questa lista viene largamente utilizzata da tecnici, hobbisti e studenti, che così condividono i loro progetti BASIC Stamp e pongono domande al riguardo.
- Stamps in Class – Creato per educatori e studenti, questa lista ha 500 sottoscrittori che discutono l'uso dei curricula Stamps in Class nei loro corsi. La lista fornisce una opportunità sia per gli studenti che per gli educatori di fare domande ed ottenere risposte.
- Parallax Educators –Esclusivamente per educatori e per coloro che contribuiscono allo sviluppo dello Stamps in Class. Parallax ha creato questo gruppo per avere un response ai nostri curricula e fornire un forum perchè gli educatori sviluppino ed ottengano supporto all'insegnamento.
- Parallax Translators – Consistente in meno di 10 persone, lo scopo di questa lista è fornire un canale tra la Parallax e coloro che traducono la nostra documentazione nelle lingue diverse dall'inglese. La Parallax fornisce documenti Word editabili ai nostri traduttori, e cerca di coordinare nel tempo le traduzioni con le pubblicazioni.
- Toddler Robot – Un cliente ha iniziato questa lista di discussione per parlare di applicativi e programmazione del robot Toddler della Parallax.
- SX Tech – Discussione circa la programmazione del microcontrollore SX con strumenti di programmazione assembler Parallax e compilatori BASIC e C di altri fornitori. Circa 600 membri.
- Javelin Stamp – Discussione degli applicativi e progetti con il Javelin Stamp, un modulo Parallax che si programma usando un sottoinsieme del linguaggio di programmazione Java della Sun. Approssimativamente 250 membri.

ERRATA

Sebbene venga fatto un grande sforzo per assicurare l'accuratezza dei nostril testi, alcuni errori possono tuttavia ancora persistere. Se trovate un errore, per favore comunicatecelo inviandoci un e-mail ad editor@parallax.com. Ci sforziamo continuamente di migliorare i nostri materiali educativi e la nostra documentazione , quindi revisioniamo in continuazione i nostri testi. Saltuariamente, un foglio di Errata con una lista di errori conosciuti, e di correzioni per un determinato testo, sarà inserito nel nostro sito, www.parallax.com. Vi preghiamo di controllare nei download gratuiti, le singole pagine di prodotto per file di errata.

Indice

Prefazione

UN ROBOT AUTONOMO ED UN TELECOMANDO

Il telecomando può essere lo strumento per fare zapping tra i canali comodamente seduti sul divano, ma questo dispositivo può anche essere usato per inviare messaggi al vostro robot Boe-Bot™. La pressione di un tasto del telecomando rivela tutta una nuova serie di possibilità per il Boe-Bot. Una volta compreso come un telecomando usa gli infrarossi per trasmettere messaggi ad un TV o ad un Videoregistratore, programmare il microcontrollore BASIC Stamp® 2 per ricevere e processare questi messaggi è molto facile. Non appena questi compiti sono trasformati in subroutine, la programmazione del Boe-Bot per azionarlo in base a questi messaggi è facile come uno schiocco di dita.

Sono di seguito elencate alcune applicazioni del Boe-Bot che avrete l'opportunità di provare nei prossimi tre capitoli:

- Premere e mantenere premuti dei tasti sul telecomando per controllare il vostro Boe-Bot come una automobile telecomandata.
- Inviare messaggi al vostro Boe-Bot mentre stà autonomamente navigando per cambiare il modello di comportamento.
- Abilitare/Disabilitare il programma del Boe-Bot con il tasto acceso/spento.
- Indicare al Boe-Bot quale programma attivare.
- Tele controllare il Boe-Bot con sequenze di movimenti.

Durante il corso, imparerete l'uso della modulazione a larghezza di impulso (PWM) per l'invio di messaggi elettronici, il sistema numerico binario, il comando PBASIC **`PULSIN`**, e nuovi usi dei comandi **`RCTIME`** e **`SELECT…CASE`**.

UDITORIO

Questo testo è organizzato in modo che possa essere usato dalla più grande varietà di studenti così come da allievi autodidatti. Gli studenti di scuole medie possono provare gli esempi di questo testo in una modalità assistita, semplicemente seguendo le istruzioni contrassegnate e con la supervisione degli istruttori. All'altro capo dello spettro, la comprensione da parte degli studenti della pianificazione e la capacità nella risoluzione dei problemi possono essere valutate con le domande, gli esercizi ed i progetti (con le soluzioni) nel sommario di ciascun capitolo. L'allievo/a autodidatta può lavorare al

proprio ritmo, ed ottenere assistenza tramite i forum dei gruppi di discussione Stamps in Class® citati in seguito.

ASSISTENZA E GRUPPI DI DISCUSSIONE

I seguenti due forum per gruppi di discussione sono a disposizione per coloro che desiderano supporto nell'uso di questo testo. Questi gruppi sono accessibili nel sito www.parallax.com alla sezione Discussion Forums nel menù Support.

Stamps In Class Group: Aperto agli studenti, agli educatori, ed agli autodidatti, questo forum permette ai membri di porre domande e condividere le risposte mentre progrediscono nelle attività, gli esercizi ed i progetti di questo testo.

Parallax Educator's Group: Questo forum moderato fornisce supporto agli educatori, sono graditi i consigli visto che continuiamo a sviluppare i nostri curriculum Stamps in Class. Per aderire a questo gruppo dovrete provare il vostro status di educatori, verificabile dalla Parallax. La guida per insegnanti per questo testo è disponibile per il download gratuito da questo forum.

Educational Support: stampsinclass@parallax.com Contattare direttamente il team Stamps in Class della Parallax se incontrate delle difficoltà nella sottoscrizione di questo forum di discussione, avete domande circa i materiali usati in questo testo, i nostri curriculum, i corsi per educatori o qualsiasi dei nostri servizi educativi.

Educational Sales: sales@parallax.com Contattare il nostro ufficio vendite per informazioni circa la politica di sconti educazionali e dei pacchetti per classi dei nostri kit Stamps in Class ed altri prodotti selezionati.

Technical Support: support@parallax.com Contattare il nostro team di supporto tecnico per domande generiche riguardanti l'assemblaggio e l'uso di qualsiasi dei nosti prodotti hardware o software.

GUIDA PER GLI INSEGNANTI

Il sommario di ciascun capitolo contiene un insieme di domande, esercizi e progetti comprensivi di soluzioni. Una guida per insegnanti è disponibile anche per questo testo. Contiene un ulteriore gruppo di domande, esercizi e progetti risolti, a volte ci sono anche soluzioni espanse o alternative per i materiali richiesti dal testo. La guida per insegnanti è disponibile gratuitamente sia in formato Word che PDF associandosi al Forum Educatori

della Parallax o può essere richiesta tramite e-mail stampsinclass@parallax.com. Per ottenere questi file, dovrete fornire la prova del vostro status di insegnanti.

LA SERIE DIDATTICA STAMPS IN CLASS

La Parallax ha creato il programma didattico Stamps in Class per soddisfare la richiesta di risorse elettroniche ed ingegneristiche a basso costo. *IR Remote for the Boe-Bot* è una sequenza della nostra guida per studenti Stamps in Class più popolare, *Robotica con il Boe-Bot*. Potete continuare I vosti studi con una qualsiasi delle nostre guide per studenti elencate di seguito. Tutti i libri elencati sono disponibili per il download gratuito dal sito www.parallax.com. Le versioni citate sotto sono quelle correnti al momento della revisione finale di questo libro. Per ottenere le ultime revisioni, controllate i nostri siti web www.parallax.com o www.stampsinclass.com; ci sforziamo continuamente di migliorare il nostro programma didattico.

Guide per studenti Stamps in Class:

Per una introduzione ben formulata alle pratiche di progettazione per i dispositivi ed i macchinari moderni, è altamente raccomandato lo studio delle attività e dei progetti delle seguenti guide per studenti. *Che cosa è un Microcontroller?* È il testo iniziale della serie, ed è quindi raccomandato prima dei titoli seguenti.

"Che cosa è un Microcontroller?"
Guida per Studenti, Versione 2.2, Parallax Inc., 2004 (Disponibile in lingua Italiana)

***"Applied Sensors"*, Student Guide, Version 1.3, Parallax Inc., 2003** (solo in Inglese)
***"Basic Analog and Digital"*, Student Guide, Version 1.3, Parallax Inc., 2004** (solo in Inglese)
***"Industrial Control"*, Student Guide, Version 1.1, Parallax Inc., 1999** (solo in Inglese)

Kit di Robotica:

Alcuni studenti iniziano il curriculum Stamps in Class con la guida per studenti *Robotica con il Boe-Bot*. Dopo averla completata, sarete pronti per i testi ed i kit di robotica seguenti:

"Robotica con il Boe-Bot"
Guida per Studenti, Versione 2.2, Parallax Inc., 2004 (Disponibile in lingua Italiana)

"Infrarossi per il controllo a distanza del Boe-Bot"
Guida per Studenti, Versione 1.0, Parallax Inc., 2004 (Questo testo)
"Advanced Robotics with the Toddler"
Student Guide, Version 1.3, Parallax Inc., 2004 (solo in Inglese)
"SumoBot", Manual, Version 2.0, Parallax Inc., 2004 (solo in Inglese)

Kit per Progetti Didattici:

Elements of Digital Logic, Understanding Signals ed *Experiments with Renewable Energy* si concentrano maggiormente su argomenti di elettronica, mentre *StampWorks* fornisce una varietà di progetti utili agli hobbisti, gli inventori ed i progettisti di prodotti interessati a provare differenti modalità progettuali.

***"Elements of Digital Logic",* Student Guide, Version 1.0, Parallax Inc., 2003** (solo in Inglese)
"Experiments with Renewable Energy"
Student Guide, Version 1.0, Parallax Inc., 2004 (solo in Inglese)
***"StampWorks",* Manual, Version 1.2, Parallax Inc., 2001** (solo in Inglese)
***"Understanding Signals",* Student Guide, Version 1.0, Parallax Inc., 2003** (solo in Inglese)

Riferimenti

"*BASIC Stamp Manual*", Version 2.0c, Parallax Inc., 2000

Questo libro è un riferimento essenziale per tutte le guide peer studenti Stamps in Class. È pieno di informazioni sulla serie di moduli a microcontrollore BASIC Stamp, sul nostro Editor BASIC Stamp, e sul nostro linguaggio di programmazione PBASIC.

TRADUZIONI IN ALTRE LINGUE

I testi educativi della Parallax possono essere tradotti in altre lingue con il nostro permesso (e-mail stampsinclass@parallax.com). Se desiderate eseguire una traduzione vogliate contattarci in modo che vi si possa fornire i documenti in formato MS Word correttamente formattati con le immagini, etc. Potete anche aderire al gruppo di discussione per traduttori della Parallax da noi mantenuto. Le istruzioni per trovare il Forum Translators della Parallax sono incluse nella sezione liste di discussione su Internet che precede l'indice.

CONTRIBUTORI SPECIALI

Il team che la Parallax ha organizzato per produrre questo testo comprende: progettazione del curriculum e letteratura tecnica; Andy Lindsay, illustrazioni; Rich Allred, progettazione grafica; Larissa Crittenden, revisione tecnica; Kris Magri, edizione tecnica; Passohanie Lindsay. Grazie anche ai clienti Parallax; Robert Ang e Sid Weaver per le loro relazioni avanzate. Stamps in Class è stato fondato da Ken Gracey, e Ken vuole ringraziare lo staff Parallax per l'ottimo lavoro che stanno facendo. **Un ringraziamento anche a Stefano Caruso per la traduzione di questo testo in Italiano**. Ciascuno e tutti i Parallaxiani che hanno contribuito a questo come a tutti i testi Stamps in Class.

Capitolo 1:
Comunicazione a distanza con gli infrarossi

PER INIZIARE

Per cominciare con l'IR Remote AppKit potete usare due documenti:

- Questo libro – vi porterà dallo stadio iniziale ad uno stadio avanzato con una modalità passo-passo.
- IR Remote AppKit Documentation – una guida rapida che in forma di inserto è contenuta nella confezione AppKit e che può essere trovata in questo libro nella Appendice A.

Questo libro è, per la maggior parte, il seguito di *Robotica con il Boe-Bot*. Segue lo stesso formato in termini di presentazione di nuovo hardware, spiegazione dei funzionamenti, e dimostrazione di nuove tecniche con il PBASIC. Eseguendo le attività, rispondendo alle domande, risolvendo gli esercizi ed i progetti, accrescerete la vostra capacità di programmazione, elettronica e robotica man mano che apprenderete circa la comunicazione ed il controllo tramite infrarossi. La conoscenza e le capacità che acquisirete vi saranno utili per i vostri progetti futuri di robotica e/o progettazione di prodotti. NOTA: Per completare l'argomento di questo testo, dovrete già avere un robot Parallax Boe-Bot™ completamente assemblato e funzionante.

La IR Remote AppKit Documentation riassume alcune delle attività di questo libro in poche pagine. È principalmente lo stretto indispensabile perchè un'entusiasta con le capacità di programmazione medio-alte del BASIC Stamp® capisca come funziona la comunicazione tramite infrarossi, e come usare gli applicativi sviluppati in questo testo. Potete trovarlo sia in forma di inserto nella confezione del Parallax IR Remote AppKit sia nella Appendice A di questo testo. NOTA: non è richiesto un robot Boe-Bot; potete usare il vostro progetto BASIC Stamp.

Questo libro e la **IR Remote AppKit Documentation** sono ambedue disponibili per il download gratuito dal sito www.parallax.com.

CONTENUTO DEL KIT

Il kit IR Remote AppKit contiene un telecomando universale simile a quello mostrato in Figura 1-1. Le attività di questo capitolo faranno uso del segnale emesso da un telecomando universale dopo che è stato configurato per controllare un televisore di marca SONY®. La maggior parte dei telecomandi universali può essere configurata per inviare messaggi ad un TV SONY. Se volete provare queste attività con un telecomando universale già in vostro possesso, o uno acquistato presso un rivenditore locale, dovrete leggere le istruzioni accluse al telecomando per sapere come configurarlo per controllare un TV SONY.

Elenco Componenti per il controllo remoto tramite infrarossi:*

(1) 020-00001 Telecomando universlae con il proprio manuale
(1) 350-00014 Rivelatore IR
(1) 150-02210 Resistenza – 220 Ω
(1) 800-00016 Ponticelli in filo rigido– bustina di 10

*Due pile alcaline AA vendute separatamente

Figura 1-1
Contenuto del kit controllo a distanza con IR

Oltre ad un Boe-Bot già montato, ci sono alcuni componenti del vostro kit originale Boe-Bot Robot (vedere la Figura 1-2). Le attività di questo capitolo faranno uso dello stesso circuito di rilevazione di infrarossi usati nei capitoli 7 ed 8 di *Robotica con il Boe-Bot*. Le Resistenze, i LED, e l'altoparlante piezoelettrico dovrebbero già essere familiari dai capitoli precedenti. Per ricevere messaggi dal telecomando è necessario solamente un

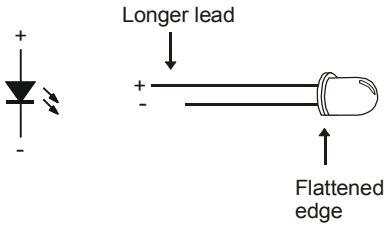
rivelatore di infrarossi. Tuttavia, alcune delle attività successive di questo capitolo faranno uso dell'intero circuito di rivelazione di IR per la navigazione autonoma combinata con il controllo remoto.

Figura 1-2
Componenti del Boe-Bot Kit

Elenco Componenti:

(2) Rivelatori di Infrarossi

(2) LED IR– (corpo chiaro)



(2) LED IR
Assieme di schermo

(2) Resistenze – 220 Ω

(2) Resistenze – 1 kΩ

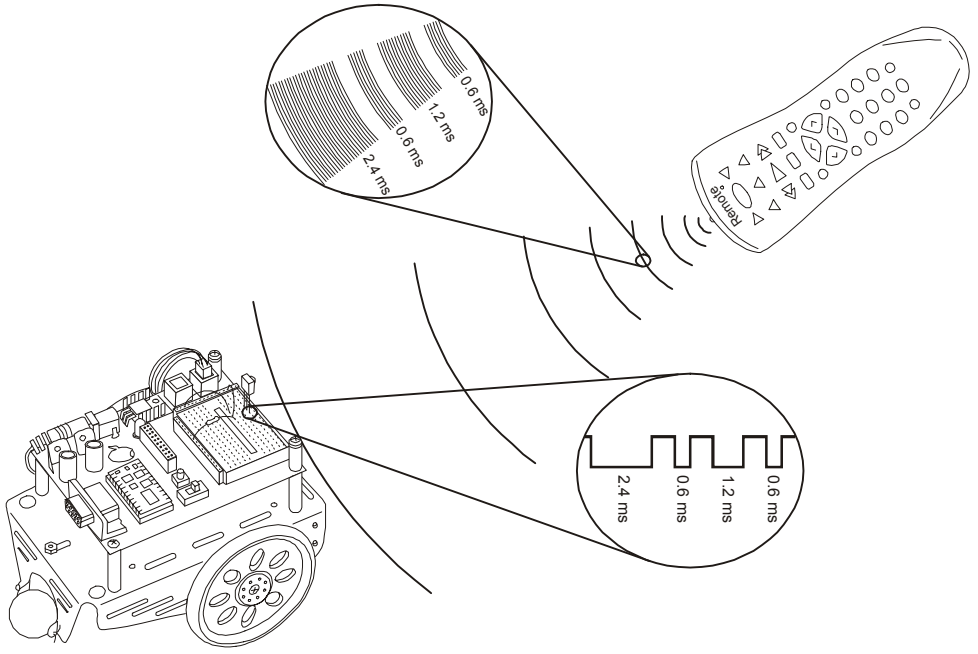
(1) Altoparlante Piezo

(2) LED – rossi

COME IL TELECOMANDO INVIA I MESSAGGI

La Figura 1-3 mostra come il telecomando può essere usato per inviare messaggi al Boe-Bot. Quando viene premuto un tasto del telecomando, esso fà lampeggiare il suo LED IR a 38.5 kHz emettendo un codice per quel tasto. Il codice viene prodotto controllando il tempo di lampeggiamento del LED IR. La sequenza di lampeggi IR emessa mostrata nella figura corrisponde all'inizio del codice che dice ad un TV SONY che è stato premuto il tasto n° 3. Per inviare messaggi ad un TV SONY, il telecomando deve emettere infrarossi per 2.4 ms segnalando che un messaggio stà per iniziare. Il messaggio che segue consiste in una combinazione di emissioni di 1.2 ms (binario-1) e di 0.6 ms (binario-0). Nota: il rivelatore IR sul Boe-Bot invia segnali bassi che corrispondono allo schema emesso dal telecomando. Il BASIC Stamp può quindi essere programmato per rilevare, misurare, memorizzare ed interpretare la durata degli impulsi bassi di questo schema per capire quale tasto del telecomando è stato premuto.

Figura 1-3: Messaggi Infrarossi emessi dal Telecomando



Il ricevitore di IR usato dal Boe-Bot per la rilevazione di oggetti tramite infrarossi in *Robotica con il Boe-Bot* è lo stesso rivelatore presente in molti televisori e videoregistratori. Questi ricevitori inviano un segnale basso ogni volta che rivelano infrarossi con una frequenza di 38.5 kHz ed un segnale alto per il resto del tempo. Quando il rivelatore IR invia dei segnali bassi, il processore all'interno del TV o del VCR misura quanto è lungo ciascun segnale basso. Quindi, usa queste misure per capire quale tasto è stato premuto sul telecomando. Come per il processore interno al TV od al VCR, il BASIC Stamp 2 può essere programmato per rivelare, misurare, memorizzare ed interpretare la sequenza di impulsi bassi che riceve dallo stesso tipo di rivelatore IR.

Modulazione a larghezza di impulso (PWM): la durata dell'impulso viene usata in molte applicazioni, alcune delle quali sono conversione da digitale ad analogico, controllo di motori e comunicazione. Quando il rivelatore IR invia degli impulsi bassi essi possono essere misurati per determinare quale informazione stà emettendo il telecomando ad infrarossi, esso è un esempio dell'uso della PWM per la comunicazione.

Segnale Portante: Il telecomando IR usa un "segnale portante" a 38.5 kHz per trasmettere la durata dell'impulso dal telecomando al rivelatore IR.

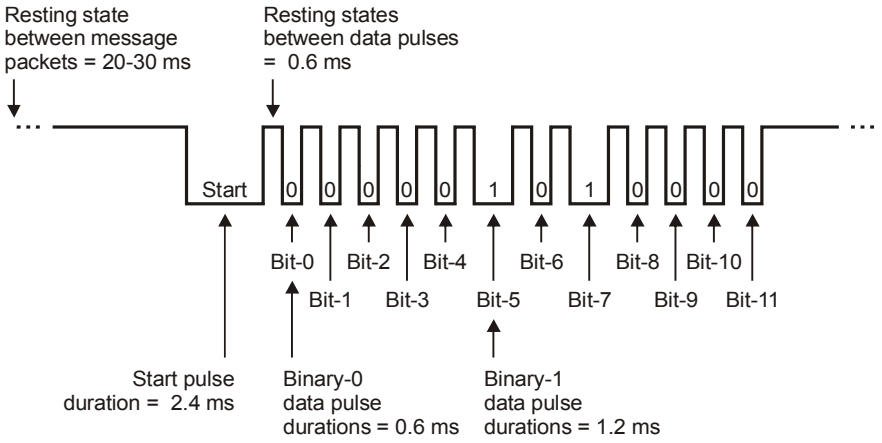
Protocollo di Comunicazione: Un protocollo di comunicazione è un insieme di regole per dispositivi che si debbono scambiare dei messaggi elettronici. I protocolli di solito hanno regole per le tensioni, il tempo di durata dei segnali, le frequenze dei segnali portanti e/o la lunghezza d'onda e molto altro. Quando due o più dispositivi seguono le regole di un dato protocollo, dovrebbero essere in grado di conunicare tra loro e scambiarsi informazioni.

Ci sono molti protocolli di comunicazione differenti che un telecomando può usare per trasmettere messaggi PWM ai componenti di un sistema multimediale. Questo testo si concentrerà sul protocollo SONY. È facile da capire, e funziona bene con lo stesso rivelatore di IR che abbiamo usato per la rilevazione degli oggetti e della loro distanza in *Robotica con il Boe-Bot*.

La Figura 1-4 mostra un esempio di diagramma di temporizzazione per un ipotetico segnale che il rivelatore di IR può inviare al BASIC Stamp quando riceve dal telecomando un messaggio di controllo con protocollo SONY TV. Questo messaggio consiste di tredici impulsi negativi che il può misurare con facilità. Il primo impulso è l'impulso di start e dura 2.4 ms. I dodici impulsi successivi potranno durare sia 1.2 ms (binario-1) che 0.6 ms (binario-0). I primi sette impulsi dei dati contengono il messaggio IR circa quale tasto è stato premuto. Gli ultimi cinque impulsi contengono un valore binario che specifica a quale periferica (TV, VCR, CD, DVD) è destinato il messaggio. Gli impulsi vengono trasmessi nell'ordine LSB, il che significa il bit meno significativo verrà trasmesso per primo cioè il primo impulso sarà bit-0, l'impulso successivo sara bit-

1, e così via. Se premete e mantenete premuto un tasto del telecomando, lo stesso messaggio verrà trasmesso in continuazione con una pausa di 20 - 30 ms tra i messaggi.

Figura 1-4: Diagramma di Temporizzazione di un Messaggio IR



In questo Capitolo, inizierete con programmi semplici per far interpretare al vostro modulo BASIC Stamp 2 l'insieme degli impulsi inviati dal telecomando e farlo agire di conseguenza. Espanderete quindi questi programmi per guidare il Boe-Bot con il telecomando ad infrarossi.

ATTIVITÀ #1: CONFIGURAZIONE DEL TELECOMANDO

In questa attività, programmerete il vostro telecomando universale in modo che esso invii messaggi PWM usando il protocollo SONY. In questo caso, il termine "programmazione" si riferisce alla sequenza di tasti che verranno premuti sul telecomando per istruirlo ad emettere impulsi secondo il protocollo SONY.

Se cambiate le pile del vostro telecomando, probabilmente dovrete ripetere I passi di questa attività. Perchè? Perchè quando rimuovete le pile dal vostro telecomando verranno persi I dati di impostazione pr il protocollo SONY.

Componenti per il telecomando ad Infrarossi

(1) telecomando Universale
(1) Manuale di istruzioni per il telecomando
(2) pile alkaline AA

Come Configurare il telecomando per il Kit Parallax

Queste istruzioni sono specifiche per il particolare telecomando fornito con il kit applicativo IR Remote AppKit. Se stete usando un telecomando diverso, controllate sul suo manuale di istruzioni per trovare il modo di programmarlo per controllare un TV SONY.

Manuali di Istruzioni mancanti: Il kit IR Remote AppKit venduto dalla Parallax è stato collaudato con queste attività, ed è completo del manuale di istruzioni per il modello di telecomando fornito. Se volete provare queste attività con un telecomando già in vostro possesso, dovrete necessariamente avere il relativo manuale di istruzioni. Se detto manuale è andato perso, potreste trovarne una copia pubblicata sul World Wide Web. I telecomandi universali possono anche essere acquistati presso i negozi di elettrodomestici locali, e la maggior parte includono i manuali di istruzioni. Controllate la confezione prima dell'acquisto di un telecomando particolare per assicurarvi che possa essere configurato per controllare un TV SONY.

Esempio: Configurazione con le Istruzioni

Queste istruzioni sono per il telecomando incluso nel kit IR Remote AppKit.

- √ Inserite due pile AA alkaline nel telecomando seguendo le istruzioni a pagina 3 del manuale di istruzioni.

Queste istruzioni sono un riassunto del contenuto di pagina 4 del manuale di istruzioni nella sezione "To Manually Program the Remote Control".

- √ Cercate SONY nella sezione “Setup Codes for Television”. Dovrebbe essere 0001 (vedere a pagina 17).
- √ Premere e rilasciare il tasto TV.
- √ Prmere e mantenere premuto il tasto SET fino a che l'indicatore LED sul telecomando si accende e rimane acceso. NOTA: IL LED può lampeggiare brevemente appena premete SET, ma dovrete mantenere premuto il tasto SET per un paio di secondi prima che il LED rimanga acceso.

√ Usate i tasti numerici per inserire 0001. Il LED potrebbe lampeggiare brevemente quando premete ciascun numero.

√ Se tutto è andato come doveva, l'indicatore LED dovrebbe spengersi e rimanere spento dopo l'inserimento dell'ultima cifra (l'uno del numero 0001).

Se il LED indicatore rimanesse acceso o emettesse alcuni lampeggi dopo che avete inserito il vostro codice: significherebbe che il codice non è stato accettato dal microcontrollore interno al telecomando; ripetete la procedura.

Il Vostro Turno – provare il telecomando su altri dispositivi

√ Se desideraste provare il telecomando sul vostro TV/VCR, etc., provate a seguire le istruzioni del telecomando per verificarne il funzionamento sulle vostre apparecchiature.

√ **Prima di affrontare la prossima attività:** se avete seguito le istruzioni del punto precedente, assicuratevi di riprogrammare il telecomando per un TV SONY!

ATTIVITÀ #2: CARATTERIZZAZIONE DEI MESSAGGI IR

Questa Attività affronta la misurazione degli impulsi che il rivelatore IR del Boe-Bot invia quando rileva messaggi dal telecomando. La Figura 1-5 mostra un diagramma di temporizzazione dell'uscita del rivelatore IR all'inizio di un messaggio IR a distanza. Il rivelatore IR invia un segnale basso quando rileva infrarossi alternati ad una frequenza di 38.5 kHz. Quando il rivelatore IR non rileva un segnale infrarosso a 38.5 kHz, invia un segnale alto. Prima che il messaggio possa venire interpretato dal modulo BASIC Stamp, le durate di questi segnali bassi dovranno essere misurate e memorizzate.

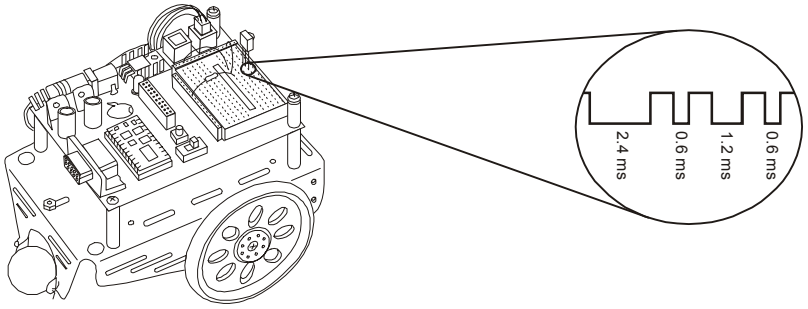


Figura 1-5
Uscita del rivelatore di IR per un messaggio remoto.

Componenti del rivelatore Infrarosso

Tutti i componenti della attività precedente
(1) Rilevatore Infrarosso
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(2) Ponticelli in filo

Il circuito di rivelazione IR

È necessario un solo rivelatore IR per ricevere messaggi dal telecomando IR (vedere la Figura 1-6).

√ Assemblate questo circuito sull'area prototipale del vostro Boe-Bot.

I numeri (1, 2, 3) sul rivelatore IR sono mostrati anche sulla mappa dei piedini in Figura 1-2. potete usarla come guida per assicurarvi che il rivelatore IR sia correttamente collegato.

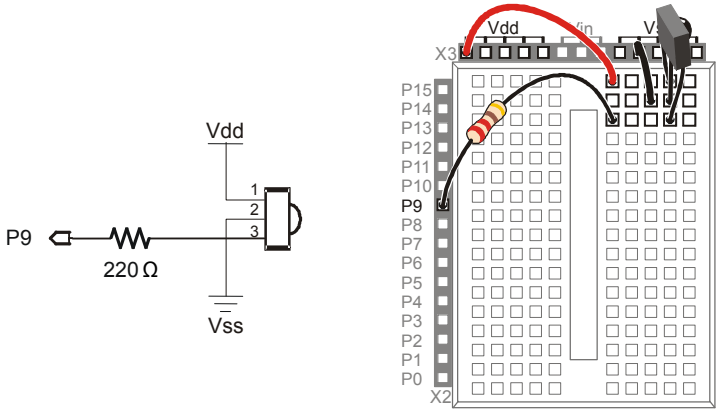


Figura 1-6 Circuito di Rilevazione IR.

Misurare gli impulsi di Start e dei Dati

Il comando **PULSOUT** che avete usato per inviare impulsi ai servo del Boe-Bot ha un comando complementare chiamato **PULSIN**. La sintassi del comando **PULSIN** è

PULSIN *Pin, State, Variable*

Pin è, chiaramente, usato per scegliere il piedino I/O su cui misurare l'impulso. ***State*** viene usato per determinare se l'impulso debba essere alto (1) o basso (0). ***Variable*** memorizza la durata misurata dal BASIC Stamp.

Impulso Alto o Impulso Basso: se la tensione ad un piedino I/O del BASIC Stamp inizialmente è bassa poi va alta per un certo tempo prima di ritornare bassa, questo è un **impulso alto**. Viene anche usato comunemente il termine **impulso positivo**. Il comando **PULSIN** misura la quantità di tempo in cui il segnale rimane alto se l'argomento ***State*** è 1.

Un **impulso basso** è l'opposto: il segnale sarà inizialmente alto, quindi andrà basso per un certo periodo di tempo prima di tornare nello stato alto. Viene anche chiamato **impulso negativo**. Il comando **PULSIN** misura la quantità di tempo in cui il segnale rimane alto se l'argomento ***State*** è 0.

Assumiamo che il vostro programma abbia una variabile di grandezza word chiamata **time** per la memorizzazione della durata misurata dell'impulso. Gli impulsi da 2.4 ms, 1.2 ms, e 0.6 ms mostrati in Figura 1-7 sono impulsi negativi. Per misurarli con il circuito rivelatore di IR, dovrete usare il comando:

```
PULSIN 9, 0, time
```

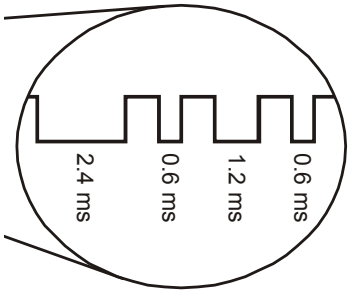


Figura 1-7
Una visione ingrandita degli Impulsi.

Il tempo alto tra due impulsi bassi può essere misurato come impulso positivo. È vero, che non contiene dati, ma può essere utile per calcolare quanto tempo impiega un messaggio IR intero ad essere trasmesso. Cambiando l'argomento ***State*** del comando **PULSIN** da 0 a 1, potete misurare la durata di un impulso positivo, come questo:

```
PULSIN 9, 1, time
```

Programma Esempio: CountStartPulses.bs2

Il primo impulso che esamineremo usando il comando **PULSIN** è l'impulso di start da 2.4 ms. Questo impulso non sarà esattamente di 2.4 ms (2400 µs), ma dovrebbe esserci abbastanza vicino, più o meno 250 µs. Questo è l'impulso che significa che altri dodici impulsi di dati verranno inviati dal telecomando.

CountStartPulses.bs2 conta il numero di impulsi bassi ricevuti la cui duraata ricade nella gamma da 1.95 a 2.85 ms. La Figura 1-8 mostra ciò che il vostro Terminale di Debug dovrebbe visualizzare dopo che avete premuto un tasto numerico per un paio di secondi puntando il telecomando verso il rivelatore IR. La durata mostrata nel Terminale di Debug per un impulso campione di start è 2562 µs. Sebbene non sia esattamente 2400 µs, è ampiamente entro i $^{+}$/- 250 µs.

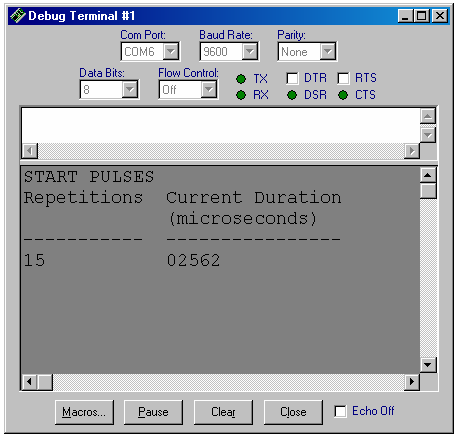


Figura 1-8
Terminale di Debug di CountStartPulses.bs2

√ Digitate e lanciate CountStartPulses.bs2.
√ Puntate il telecomando verso la lente del rivelatore di infrarossi.
√ Premere e mantenere premuto uno dei tasti numerici (0 – 9) sul telecomando.
√ Assicuratevi che vengano rilevati impulsi nella gamma 2.25 - 2.75 ms (il Terminale di Debug visualizzerà valori da 2250 a 2750 µs).
√ Annotate quì: _____ il valore effettivo dell'impulso di start del vostro telecomando. Se il valore fosse ogni volta diverso, calcolate il valore medio.

Se il programma non rileva l'impulso di start, assicuratevi che il vostro telecomando sia impostato per controllare un TV SONY. Per prima cosa, premete il tasto TV button sul telecomando, quindi provate ad inviare un altro messaggio puntando verso il vostro BASIC Stamp. Se questo non dovesse funzionare, ripetete le procedure dell' Attività #1. Se queste due soluzioni non funzionano, controllate i collegamenti del vostro circuito di rivelazione IR, cercate eventuali errori nel vostro programma. Ancora sfortunati? Provate a configurare il telecomando per controllare un TV SONY usando le istruzioni nel manuale del telecomando.

Non premete i tasti CBL, VCR, o TV/VCR. Se premete uno di questi tasti, ordinate al telecomando di inviare messaggi ad un dispositivo diverso, che sia un VCR o un decoder. In ogni caso, i messaggi che il telecomando invierà useranno un protocollo TV non SONY. Il risultato sarà che i programmi di questo libro sembrerà che non funzionino.

Se per sbaglio premeste uno di questi tasti: semplicemente premete il tasto TV per far ritornare il telecomando ad emettere segnali TV (SONY).

```
' IR Remote for the Boe-Bot - CountStartPulses.bs2
' Capture and count the number of 2.4 ms (low) start pulses.

' {$STAMP BS2}
' {$PBASIC 2.5}

time              VAR     Word
counter           VAR     Word

DEBUG "START PULSES", CR,
      "Repetitions  Current Duration", CR,
      "             (microseconds)  ", CR,
      "-----------  ----------------"

DO

  PULSIN 9, 0, time

  IF (time > 975) AND (time < 1425) THEN

    counter = counter + 1

    DEBUG CRSRXY, 0, 4,
          DEC counter,
          CRSRXY, 13, 4,
          DEC5 time * 2

  ENDIF

LOOP
```

Come funziona CountStartPulses.bs2

Questo programma inizia dichiarando due variabili word, **`time`** per memorizzare la durata dell'impulso di start, e **`counter`** per memorizzare il numero di impulsi di start ricevuti. Quindi, un comando **`DEBUG`** aggiunge alcune intestazioni di colonna per la visualizzazione di variabili.

```
time              VAR     Word
counter           VAR     Word

DEBUG "START PULSES", CR,
      "Repetitions  Current Duration", CR,
      "             (microseconds)  ", CR,
      "-----------  ----------------"
```

All'interno del ciclo **`DO…LOOP`**, il programma esegue il comando **`PULSIN 9, 0, time`**, che misura gli impulsi. Ogniqualvolta la durata di un impulso è tra 975 (1.95 ms) e 1425

(2.85 ms), il blocco di codice **IF…THEN…ENDIF** incrementa la variabile **counter** e visualizza le variabili **counter** e **time** sotto le intestazioni **Repetitions** e **Current Duration**.

```
DO

  PULSIN 9, 0, time

  IF (time > 975) AND (time < 1425) THEN

    counter = counter + 1

    DEBUG CRSRXY, 0, 4,
          DEC counter,
          CRSRXY, 13, 4,
          DEC5 time * 2

  ENDIF

LOOP
```

Che cosa significa il * 2 nel comando DEBUG? Ricordate che il comando **PULSIN** misura la durata in unità di 2 µs. Questo è ciò che verrà memorizzato nella variabile **time**, il numero di unità di 2 µs che il comando **PULSIN** ha misurato. moltiplicando **time** per 2 con l'operatore *****, il comando **DEBUG** visualizza il reale numero di microsecondi della misura dell'impulso.

Il Vostro Turno – Misurare gli impulsi Binary-1 e Binary-0

Modificando le dichiarazioni **DEBUG** ed **IF…THEN** nel programma esempio, potrete rilevare e visualizzare la durata degli impulsi binary-1 e binary-0. Ecco come ottenerlo:

- √ Salvate CountStartPulses.bs2 con il nome MeasureBinary1Pulses.bs2.
- √ Cambiare

```
DEBUG "START PULSES", CR,
```

in

```
DEBUG "BINARY-1 PULSES", CR,
```

- √ Cambiare

```
IF (time > 975) AND (time < 1425) THEN
```

in

```
IF (time > 450) AND (time < 750) THEN
```

√ Salvate il vostro programma modificato.
√ Lanciate il programma.
√ Premere e mantenere premuto uno dei tasti numerici fino a che il Terminale di Debug non visualizza la durata di un impulso.
√ Annotare la durata di binary-1 quì: _____.
√ Salvate MeasureBinary1Pulses.bs2 con il nome MeasureBinary0Pulses.bs2.
√ Cambiare

```
DEBUG "BINARY-1 PULSES", CR,
```

in

```
DEBUG "BINARY-0 PULSES", CR,
```

√ Cambiare

```
IF (time > 450) AND (time < 750) THEN
```

in

```
IF (time > 150) AND (time < 450) THEN
```

√ Salvate il vostro programma modificato.
√ Lanciate il programma ed usate uno dei tasti numerici del telecomando per inviare messaggi al vostro Boe-Bot.
√ Annotare la durata di binary-0 quì: _____.

Misurare lo Stato di Riposo tra Messaggi

Quando premete e mantenete un dato tasto sul telecomando, il telecomando invia il codice di quel tasto, quindi attende per un periodo ed invia di nuovo il codice. CountStartPulses.bs2 può anche essere modificato per conoscere la durata di questo intervallo tra messaggi. La durata di questo intervallo è mostrata in Figura 1-9, e si è rivelata essere un fattore importante nella navigazione del Boe-Bot.

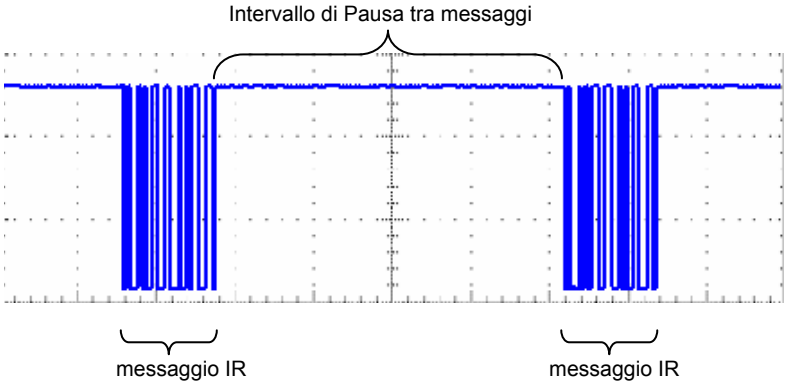


Figura 1-9
Due messaggi IR

Questa schermata dell'Oscilloscopio USB della Parallax mostra due messaggi IR separati da una pausa.

Per misurare lo stato di intervallo con il modulo BASIC Stamp, tutto ciò che dovete fare è pensare all'intervallo con segnale alto tra i messaggi come un impulso positivo con una durata molto lunga. Il comando **`PULSIN`** deve essere modificato per rilevare un impulso positivo cambiando l'argomento ***`State`*** da 0 ad 1. Anche l'istruzione **`IF…THEN`** dovrà essere modificata in modo che non esegue acun codice a meno che il tempo misurato sia inferiore a 1000 (2 ms). Questo ci assicura che i brevi impulsi alti tra l'impulso di start e gli impulsi dei dati non influenzino la misura. Invece, verranno misurati, solamente gli impulsi alti più lunghi, quelli tra i messaggi.

Rogamma Esempio: CountRestingStates.bs2

- √ Digitate e lanciate CountRestingStates.bs2.
- √ Puntate il telecomando verso il rivelatore IR, premete e mantenete premuto il tasto 5.
- √ Assicuratevi che vengano rilevati impulsi alti nella gamma da 20 a 35 ms (20,000 to 35,000 µs).
- √ Annotate quì: _____la durata dell'intervallo tra messaggi.

```
' IR Remote for the Boe-Bot - CountRestingStates.bs2
' Capture and count the number of 20 ms+ (high) resting states.

'{$STAMP BS2}
'{$PBASIC 2.5}

time        VAR     Word
counter     VAR     Word

DEBUG "RESTING STATE", CR,
      "Repetitions  Current Duration", CR,
      "             (microseconds)  ", CR,
      "-----------  ----------------"

DO

  PULSIN 9, 1, time

  IF (time > 1000) THEN

    counter = counter + 1

    DEBUG CRSRXY, 0, 4,
          DEC counter,
          CRSRXY, 13, 4,
          DEC5 time * 2

  ENDIF

LOOP
```

Come Funziona CountRestingStates.bs2

CountRestingStates.bs2 è semplicemente CountStartPulses.bs2 con alcune modifiche. La prima modifica era semplicemente l'intestazione nella finestra del Terminale di Debug. Il comando **`DEBUG "START PULSES", CR,`** è stato cambiato in **`DEBUG "RESTING STATE", CR,`**. Quindi, il programma deve cercare impulsi alti della durata da 20 a 35 ms invece di impulsi bassi che durano appena 2.4 ms. Per misurare gli impulsi alti invece degli impulsi bassi, il comando **`PULSIN 9, 0, time`** è stato cambiato in **`PULSIN 9, 1, time`**. Per misurare durate di 20-35 ms invece di durate di 2.4 ms, la condizione del blocco di codice **`IF…ENDIF`** è stata cambiata
da: **`IF (time > 975) AND (time < 1425) THEN`**
in: **`IF (time > 1000) THEN`**.

Il Vostro Turno – Misurare l'intervallo tra gli impulsi dei dati

Potete modificare ancora la condizione del blocco **IF…ENDIF**, questa volta per misurare gli impulsi alti, molto brevi, tra gli impulsi dei dati.

- √ Salvate CountRestingStates.bs2 con il nome CountRestingStatesYourTurn.bs2.
- √ Cambiate la condizione del blocco di codice **IF…ENDIF** da

```
IF (time > 1000) THEN
```

a

```
IF (time > 1) AND (time < 1000) THEN
```

- √ Aaggiungete un comando **PAUSE 100** subito prima del comando **LOOP**.
- √ Lanciate il programma ed annotate quì: _____l'intervallo tra gli impulsi.

Diagramma di Temporizzazione dei messaggi IR

La Figura 1-10 mostra un diagramma di temporizzazione di un messaggio IR proveniente da un telecomando. Questo diagramma mostra la temporizzazione mentre il tasto 5 della tastiera del telecomando viene premuto e mantenuto premuto. Il vostro compito sarà di compilare la lista delle misure con i dati che avete annotato in questa Attività. Ricordate che ci sono 1000 µs in ciascun millisecondo. Le vostre misure sono state effettuate in microsecondi, quindi dovrete dividere ciascuna misura per 1000 per inserire le misure in millisecondi nel diagramma di temporizzazione.

- √ Usate le misurazioni che avete annotato in questa Attività per ottenere le misure in millisecondi nella Figura 1-10.
- √ Asommata tutti i tempi per calcolare il tempo totale del messaggio: un intervallo tra i messaggi, un impulso di start, dodici intervalli tra gli impulsi dei dati, dodici impulsi dei dati, due dei quali sono binary-1 ed i dieci restanti sono binary-0.
- √ Annotate quì: _____la durata del vostro messaggio IR.

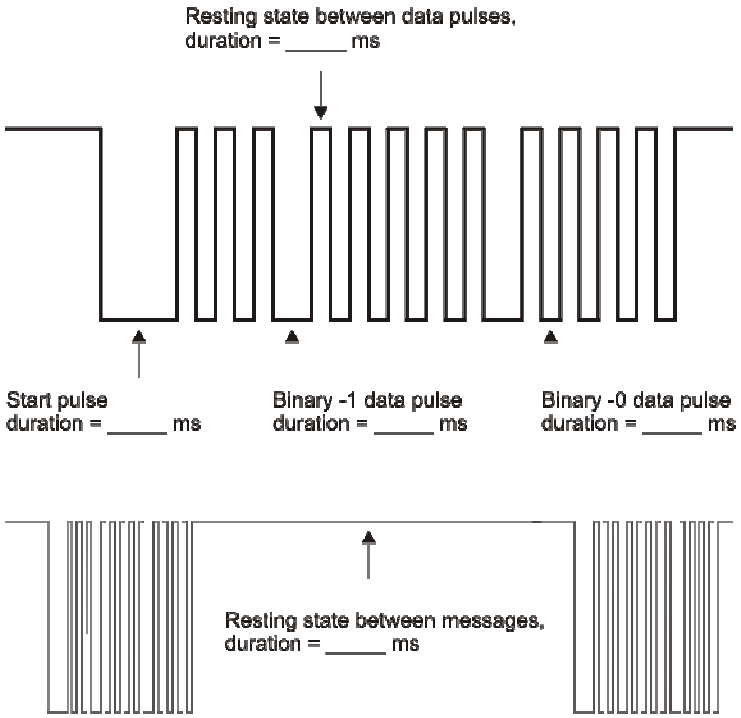


Figura 1-10
Diagramma di Temporizzazione di un messaggio IR.

Annotate le vostre misurazioni dei tempi indicati nel diagramma.

Il Vostro Turno – Conteggio del Numero di Messaggi al Secondo

Premendo e mantenendo premuto un tasto del telecomando per dieci secondi, potete determinare quanti messaggi al secondo invia il telecomando.

- √ Lanciate ancora una volta CountStartPulses.bs2; era il primo programma esempio di questa Attività.
- √ Con l'aiuto di un orologio o di un cronometro premete e mantenete premuto il tasto 5 per dieci secondi.
- √ Dividete il numero dei messaggi per 10.
- √ Annotate quì _____ il numero dei messaggi al secondo.
- √ Potete usare la formula seguente per calcolare il tempo impiegato per l'emissione di un messaggio IR e di un intervallo tra messaggi:

$$packet\_time = \frac{1}{messages / second}$$

√ Confrontate questa durata di messaggio IR con quella che avete calcolato sommando tutte le componenti del messaggio IR.

ATTIVITÀ #3: ACQUISIZIONE DEI MESSAGGI IR

In questa Attività, scriverete programmi per misurare ed acquisire ciascun impulso del messaggio PWM inviato dal telecomando ad infrarossi. In ogni messaggio ci sono un totale di dodici impulsi di dati, e ciascuno può essere acquisito e memorizzato. Programmando il BASIC Stamp per acquisire e memorizzare questi impulsi, avrete l'ingrediente base per l'invio di messaggi al vostro Boe-Bot con il telecomando.

Introduzione alle Variabili Array

Le misure di dodici impulsi devono essere memorizzate separatamente perchè essi costituiscono ciascuno una componente unica del messaggio. Ma anche così, essi sono correlati tra di loro. Sono tutte misure di impulsi, e devono essere esaminati per determinare quale tasto sul telecomando è stato premuto.

Il modo migliore per memorizzare un gruppo di valori correlati è con un tipo particolare di variabile chiamata variabile array. Una variabile array è un gruppo di variabili che hanno tutte lo stesso nome. Dichiarare una variabile array è simile alla dichiarazione di molte altre variabili. La sola differenza è che viene aggiunto tra parentesi un numero vicino all'argomento ***Size*** che indica quante variabili dovranno essere create che condivideranno lo stesso nome. Questa è la sintassi della dichiarazione di una variabile array:

```
name   VAR  Size(n)
```

questo è un esempio di un array che può memorizzare 5 byte:

```
caratteri VAR Byte(5)
```

La Figura 1-11 mostra la variabile di cinque **caratteri** creata da questa dichiarazione: **caratteri(0)**, **caratteri(1)**, **caratteri(2)**, **caratteri(3)**, e **caratteri(4)**. Ciascuna di queste variabili, chiamate elementi dell' array, può memorizzare un byte. Ciascuna di queste variabili ha un numero chiamato indice (il numero tra parentesi) che la distingue dagli altri elementi. In altre parole, eciascun elemento dell' array ha lo stesso nome, ma un diverso numero indice.

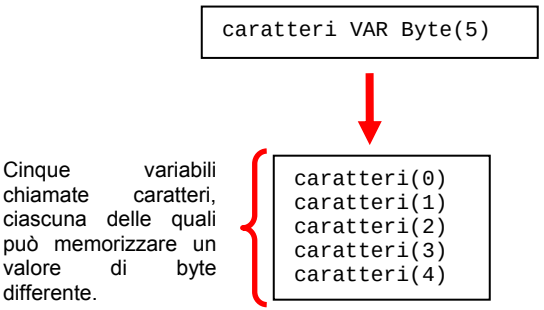


Figura 1-11
Elementi in un Array

Progamma Esempio: ArrayEsempio.bs2

Questo pProgramma Esempiousa il Terminale di Debug per impostare gli elementi dell' array e visualizzarne il contenuto. La Figura 1-12 mostra un esempio di come apparirà il Terminale di Debug dopo che avrete lanciato e provato il programma. Ciascun carattere che digiterete nella finestra di trasmissione del Terminale di Debug viene caricato nell'elemento successivo dell'array **`caratteri`**. Dopo che avrete digitato il quinto carattere, il programma legge e visualizza ciascun elemento dell'array.

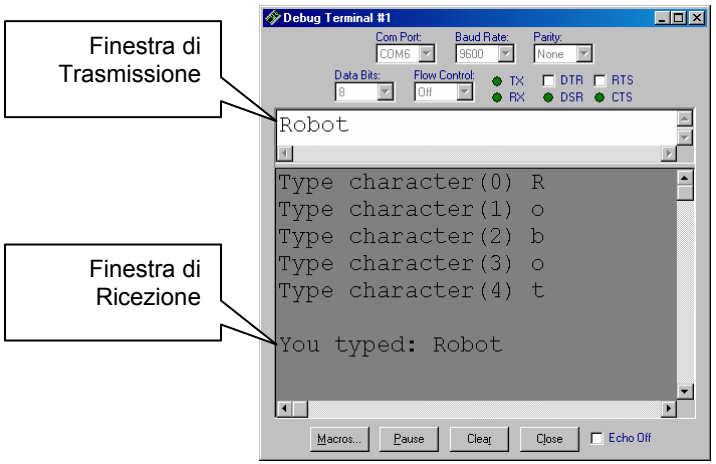


Figura 1-12
Uso del Terminale di Debug per digitare Caratteri in un Array

Clickare sulla finestra di trasmissione, quindi digitare cinque caratteri.

- √ Digitate e lanciate ArrayEsempio.bs2.
- √ Clickate la finestra di tasmissione del Terminale di Debug.
- √ Digitate cinque caratteri.
- √ Quando il Terminale di Debug visualizza "You typed: …", verificate che il BASIC Stamp abbia inviato gli stessi caratteri che avete digitato.

```
' IR Remote for the Boe-Bot - ArrayEsempio.bs2
' Set array element values with DEBUGIN and display them with DEBUG.

' {$STAMP BS2}
' {$PBASIC 2.5}

characters     VAR     Byte(5)
index          VAR     Nib

FOR index = 0 TO 4
  DEBUG "Type character(", DEC index, ") "
  DEBUGIN characters(index)
  DEBUG CR
NEXT

DEBUG CR, "You typed: "

FOR index = 0 TO 4
  DEBUG characters(index)
NEXT

END
```

Come Funziona ArrayEsempio.bs2

Viene dichiarato un array di cinque byte chiamato **`characters`** insieme con una variabile nibble chiamata **`index`**.

```
characters     VAR     Byte(5)
index          VAR     Nib
```

un ciclo **`FOR…NEXT`** ripete cinque volte il blocco di codice tra le istruzioni **`FOR`** e **`NEXT`**. A ciascuna iterazione del ciclo, il valore di **`index`** viene incrementato di 1, in questo modo il comando **`DEBUGIN`** memorizza il carattere nell' elemento successivo dell'array.

```
FOR index = 0 TO 4
  DEBUG "Enter character(", DEC index, ") "
  DEBUGIN characters(index)
  DEBUG CR
NEXT
```

Un secondo ciclo **FOR…NEXT** legge e visualizza i valori di ciascun elemento dell' array **characters**.

```
FOR index = 0 TO 4
  DEBUG characters(index)
NEXT
```

Alla prima iterazione del ciclo **FOR…NEXT**, il valore di **index** è 0, quindi viene visualizzato **characters**(0). Alla seconda iterazione, **index** è 1, quindi viene visualizzato **characters**(1), e così via.

Il Vostro Turno – Memorizzazione di Va<lori invece che Caratteri

Potete usare il modificatore **DEC** per memorizzare valori invece che caratteri. Ecco come:

- √ Salvate ArrayEsempio.bs2 con il nome ArrayEsempioYourTurn.bs2.
- √ Cambiate

```
DEBUG "Type character(", DEC index, ") "
DEBUGIN characters(index)
DEBUG CR
```

in

```
DEBUG "Type value - character(", DEC index, ") "
DEBUGIN DEC characters(index)
```

- √ Cambiate

```
DEBUG characters(index)
```

in

```
DEBUG CR, DEC characters(index)
```

- √ Lanciate il programma modificato e digitate valori (da 0 a 255). Assicuratevi di premere il tasto Enter dopo l'ultima cifra di ciascun valore che inserite.
- √ Verificate che il programma visualizzi i valori che avete memorizzato in ciascun elemento dell'array.

acquisizione dell'intero messaggio

Un messaggio completo del telecomando ha dodici impulsi di dati. Ecco come acquisire tutte le reispettive durate:

- Continuate ad eseguire il comando **PULSIN** fino a che non viene rilevato l'intervallo tra gli impulsi.
- Usate altri dodici comandi **PULSIN** per acquisire i successivi dodici impulsi di dati e memorizzarli in un array di variabili word.
- Opzionale – usate un ciclo **FOR…NEXT** per visualizzare gli elementi dell' array.

La Figura 1-13 mostra la dichiarazione della variabile array **time VAR Word(12)**. La dichiarazione crea dodici diverse variabili word, ciascuna delle quali può memorizzare il proprio valore. Nel caso della dichiarazione dell' array di variabile **time**, esistono **time(0)**, **time(1)**, **time(2)**, e così via, fino a **time(11)**. Ciascuno degli elementi dell'array **time** può memorizzare un valore diverso tra 0 e 65535.

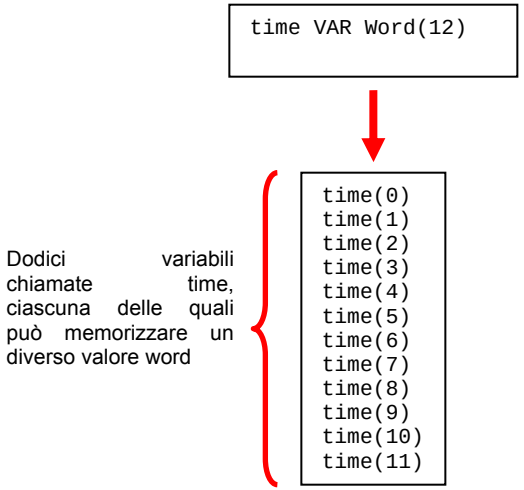


Figura 1-13 Elementi nell'array di variabili **time.**

Il prossimo pProgramma Esempionon userà un ciclo per impostare il valore di ciascun elemento dell' array **time**. A causa del fatto che il tempo tra gli impulsi inviati dal telecomando è molto breve, il codice aggiuntivo di un ciclo **FOR…NEXT** può causare la perdita di alcuni impulsi. Quindi ciascun elemento dell'array verrà impostato sequenzialmente da dodici comandi **PULSIN**.

```
PULSIN 9, 0, time(0)
PULSIN 9, 0, time(1)
PULSIN 9, 0, time(2)
  •
  •
  •
PULSIN 9, 0, time(11)
```

Ogni volta che vedrete tre punti tra i comandi, significa che ci sono altri elementi nella sequenza che sono stati omessi per semplificare la spiegazione. Per esempio, nella sequenza di comandi **PULSIN**, i

•
•
•

Indicano che esistono ma non sono mostrati i comandi da **PULSIN 9, 0, time(3)** fino a **PULSIN 9, 0, time(10)**. Tre punti **...** sono anche usati per descrivere comandi come **FOR…NEXT**, **DO…LOOP**, ed altre dichiarazioni che hanno blocchi di codice tra l'inizio e la fine delle parole chiave. I tre punti **...** indicano che ci sono uno o più comandi tra le parole chiave che non sono stati mostrati.

Progamma Esempio: RecordAndDisplayPwm.bs2

questo pProgramma Esempiomisura e visualizza la durata di tutti i dodici impulsi dei dati. La Figura 1-14 mostra un esempio di che cosa verrebbe visualizzato quando il tasto 5 del telecomando viene premuto e mantenuto premuto. Potete usare questo programma per esaminare lo schema degli impulsi per ciascun tasto del vostro telecomando. Ricordate che alcuni tasti non sono per il TV, come ad esempio i tasti play e pause, che funzionerebbero con un VCR. Questi tasti non fanno emettere messaggi dal telecomando quando è configurato per funzionare come controllore di un TV SONY.

Ricordate di non premere i tasti VCR, TV/VCR, o CBL. Se lo fate per errore, premete il tasto TV per ritornare alla modalità controllo TV SONY.

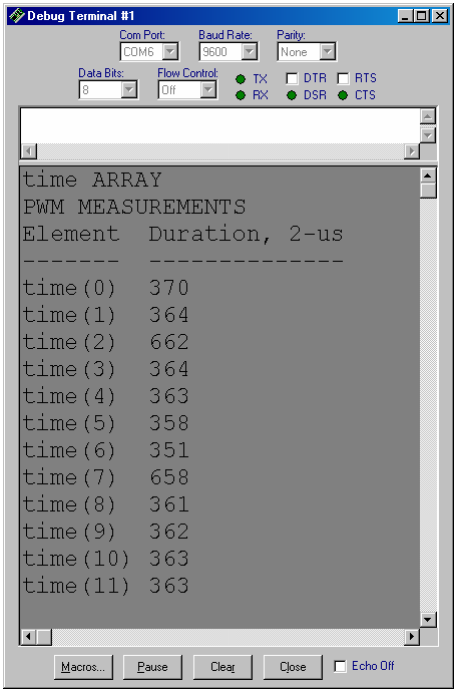


Figura 1-14
Terminale di Debug per RecordAndDisplayPwm.bs2

Elenco degli impulsi del tasto 5.

- √ Digitate e lanciate RecordAndDisplayPwm.bs2.
- √ Ridimensionate il Terminale di Debug in modo che sia alto quanto il vostro monitor, perchè altrimenti i vostri dati possono non entrarci tutti.
- √ Provate a premere e mantenere premuti i seguenti tasti e verificate che ciascun gruppo binario sia diverso per ciascun tasto: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, VOL-, VOL+, CH-, CH+, POWER, e MENU. Ponete particolare attenzione alle variabili da **`time(0)`** a **`time(6)`** e provate a rilevare lo schema delle durate per ciascun tasto.

```
' IR Remote for the Boe-Bot - RecordAndDisplayPwm.bs2
' Measure all data pulses from SONY IR remote set to control a TV.

' {$STAMP BS2}
' {$PBASIC 2.5}

time            VAR     Word(12)                ' SONY TV remote variables.
index           VAR     Nib
                                                ' Display heading.
DEBUG "time ARRAY", CR,
```

```
      "PWM MEASUREMENTS", CR,
      "Element  Duration, 2-us", CR,
      "-------  --------------"

DO                                          ' Beginning of main loop.

  DO                                        ' Wait for rest between messages.
    PULSIN 9, 1, time(0)
  LOOP UNTIL time(0) > 1000

  PULSIN 9, 0, time(0)                      ' Measure/store data pulses.
  PULSIN 9, 0, time(1)
  PULSIN 9, 0, time(2)
  PULSIN 9, 0, time(3)
  PULSIN 9, 0, time(4)
  PULSIN 9, 0, time(5)
  PULSIN 9, 0, time(6)
  PULSIN 9, 0, time(7)
  PULSIN 9, 0, time(8)
  PULSIN 9, 0, time(9)
  PULSIN 9, 0, time(10)
  PULSIN 9, 0, time(11)

  FOR index = 0 TO 11                       ' Display 12 pulse measurements.

    DEBUG CRSRXY, 0, 4 + index, "time(", DEC index, ")",
          CRSRXY, 9, 4 + index, DEC time(index)

  NEXT

LOOP                                        ' Repeat main loop.
```

come funziona RecordAndDisplayPwm.bs2

La dichiarazione della variabile **`time`** crea un array con dodici elementi word. Il programma usa anche una variabile nibble chiamata **`index`**.

```
time         VAR     Word(12)
index        VAR     Nib
```

un singolo comando **`DEBUG`** con molto testo ed argomenti crea l'intestazione delle colonne per le informazioni che devono essere visualizzate.

```
DEBUG "time ARRAY", CR,
      "PWM MEASUREMENTS", CR,
      "Element  Duration, 2-us", CR,
      "-------  --------------"
```

Il blocco di codice in questo ciclo **DO…LOOP** continua a misurare gli impulsi alti (positivi) fino a trovarne uno maggiore di 2 ms, il che significa che deve trattarsi dell'intervallo tra gli impulsi del messaggio IR.

```
DO
  PULSIN 9, 1, time(0)
LOOP UNTIL time(0) > 1000
```

Quando il comando **PULSIN** termina la misura dell'impulso di intervallo ed il programma stà decidendo cosa fare successivamente, il telecomando stà inviando l'impulso di start. Questo è il momento giusto per attivare dodici comandi **PULSIN** di seguito per acquisire i dodici dati degli impulsi. Notare come ciascuno venga memorizzato in elementi diversi dell'array. Questi dodici comandi **PULSIN** dimostrano anche come possono essere usati valori costanti come ad es. 0, 1, 2 fino ad 11 per indicizzare gli elementi di un array.

```
PULSIN 9, 0, time(0)
PULSIN 9, 0, time(1)
PULSIN 9, 0, time(2)
  •
  •
  •
PULSIN 9, 0, time(11)
```

Il prossimo è un altro esempio di come indicizzare l'indice dell' array in un altro modo, con una variabile. In questo caso, la variabile è **index**, e viene incrementata a ciascuna iterazione di un ciclo **FOR…NEXT**. L'ultimo argomento del comando **DEBUG** è **DEC time(index)**. Dal momento che il valore di **index** aumenta di uno ad ogni iterazione del ciclo **FOR…NEXT**, il comando **DEBUG** visualizza il valore memorizzato in ciascun elemento successivo dell' array **time**. Alla prima iterazione del ciclo, il comando **DEBUG** visualizza **time(0)**, alla seconda iterazione, visualizza **time(1)**, e così via.

```
FOR index = 0 TO 11

  DEBUG CRSRXY, 0, 4 + index, "time(", DEC index, ")",
        CRSRXY, 9, 4 + index, DEC time(index)

  NEXT

LOOP
```

L'ultimo comando **`LOOP`** invia il programma indietro al primo comando **`DO`**. Il blocco di codice inserito in questo **`DO…LOOP`** esterno si ripete continuamente, processando i messaggi man mano che il telecomando li invia.

Il Vostro Turno – Acquisizione della mappa degli impulsi per ciascun tasto

Le due Attività seguenti forniranno programmi che renderanno le decisioni di navigazione dettate dalla durata degli impulsi memorizzata nell' array **`time`**. I primi sette elementi dell' array **`time`** contengono tutte le misure degli impulsi che vi serviranno per identificare i diversi tasti del telecomando. La Table 1-1 ha righe per ciascuna delle sette misure dell'array **`time`** e colonne per ciascun tasto numerico e per alcuni altri tasti. Compilatela in modo che la possiate usare come riferimento mentre scrivete programmi delle due prossime attività.

- √ Premere e rilasciare ciascun tasto elencato nella Table 1-1.
- √ Compilate la colonna di ciascun tasto con le misure dell'array **`time`** prese dal Terminale di Debug.

Table 1-1: Misura del Tempo per Ciascun Tasto										
Elemento dell'Array	**Tasto del Telecomando**									
	1	2	3	4	5	6	7	8	9	0
time(0)										
time(1)										
time(2)										
time(3)										
time(4)										
time(5)										
time(6)										

Elemento dell'Array	**Tasto del Telecomando**					
	VOL-	VOL+	CH-	CH+	ENTER	POWER
time(0)						
time(1)						
time(2)						
time(3)						
time(4)						
time(5)						
time(6)						

ATTIVITÀ #4: NAVIGAZIONE A DISTANZA DEL BOE-BOT CON IR

Una applicazione piuttosto divertente è la programmazione del Boe-Bot in modo che ne possiate controllare il movimento direttamente con il telecomando, esattamente come con una macchina radiocomandata. In questa Attività, potrete programmare il Boe-Bot per riconoscere i tasti 1, 2, 3, o 4 quando li premete e li tenete premuti, per effettuare una manovra diversa per ciascuno. Potete anche usare i tasti CH+/- e VOL+/-.

Circuito e Componenti del controllo con Telecomando ad Infrarossi

Tutti i componenti delle Attività #1 e #2
(1) Boe-Bot
(1) Altoparlante Piezo
(vari) Ponticelli di filo

√ Assemblate/riassemblate e collaudate i circuiti dei servo del Boe-Bot, il rivelatore IR, ed dell' altoparlante piezo mostrati nella Figura 1-15 come mostrato.

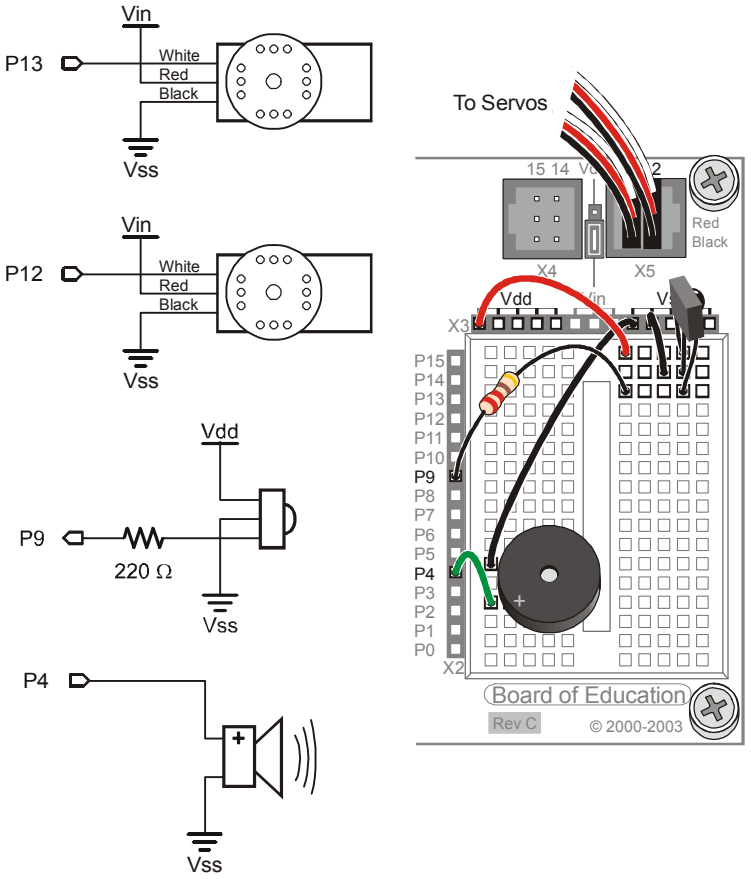


Figura 1-15
Schemi elettrici e diagrammi di collegamento dei circuiti Servo, Rivelatore IR, ed altoparlante piezo.

Il servo di sinistra del Boe-Bot deve essere collegato a P13, ed il suo servo di destra deve essere collegato a P12.

Ritorno alle Istruzioni IF…THEN per la Navigazione

Il prossimo programma esempio è una versione modificata di RecordAndDisplayPwm.bs2. ecco che cosa si deve fare al programma per fargli controllare il Boe-Bot:

- √ Rimuovoere la dichiarazione della variabile **index**.
- √ Riducete la dimensione dell'array **time** a due elementi.
- √ Aggiungere **FREQOUT 4, 2000, 3000** prima del primo **DO**.
- √ Rimuovere tutti i comandi **DEBUG**.
- √ Rimuovere tutti i comandi **PULSIN** eccetto quelli che memorizzano le misure di tempo in **time(0)** e **time(1)**.
- √ Rimuovere il ciclo **FOR…NEXT** che visualizza la durata degli impulsi.
- √ Usate le informazioni degli impulsi nella Table 1-1 per creare le dichiaraioni **IF…THEN** che scelgono gli argomenti ***Duration*** di **PULSOUT** per una manovra basata su quattro possibili combinazioni di valori che **time(0)** e **time(1)** possono memorizzare.

Tornate alla Table 1-1 e date un'occhiata alle misure memorizzate nell'array **time** per i tasti da 1 a 4. Osservando **time(1)** e **time(0)**. Se confrontate le misure degli impulsi a 500 esistono solamente quattro possibili combinazioni:

(1) Sia **time(1)** che **time(0)** sono minori di 500.
(2) **time(1)** è minore di 500, ma **time(0)** è maggiore di 500.
(3) **time(1)** è maggiore di 500, ma **time(0)** è minore di 500.
(4) Sia **time(1)** che **time(0)** sono maggiori di 500.

Queste decisamente sono informazioni sufficienti per scrivere una dichiarazione **IF…THEN** simile a quelle che guidavano il Boe-Bot con la navigazione tattile e con gli infrarossi:

```
IF (time(1) < 500) AND  (time(0) < 500) THEN
  PULSOUT 13, 850
  PULSOUT 12, 650
ELSEIF (time(1) < 500) AND (time(0) > 500) THEN
  PULSOUT 13, 650
  PULSOUT 12, 850
ELSEIF (time(1) > 500) AND (time(0) < 500) THEN
  PULSOUT 13, 850
  PULSOUT 12, 850
ELSEIF (time(1) > 500) AND (time(0) > 500) THEN
  PULSOUT 13, 650
  PULSOUT 12, 650
ENDIF
```

Programma Esempio– 2BitRemoteBoeBot.bs2

Potete premere e mantenere premuto i tasti 1, 2, 3, o 4 per selezionare una di quattro direzioni: avanti, indietro, ruotare a destra, ruotare a sinistra. Queste direzioni

funzionano anche con i tasti CH+, CH-, VOL+, e VOL-, che sono forse più adatti in quanto questi tasti puntano nelle direzioni avanti, indietro, destra e sinistra sulla maggior parte dei telecomandi.

Questi programmi sono stati scritti per i Boe-Bot con i servo a rotazione continua della Parallax. Se il vostro Boe-Bot è etichettato con le lettere "PM" evidenziate in blu, dovrete cambiare gli argomenti ***Duration*** di **PULSOUT** nei vostri programmi. Se avete i servo Parallax PM, usate 500 al posto di 650 e 1000 al posto di 850.

- √ Digitate, salvate e lanciate 2BitRemoteBoeBot.bs2.
- √ Assicuratevi che l'interruttore di alimentazione sulla Board of Education sia in posizione 2.
- √ Puntando il telecomando verso il Boe-Bot, premere e mantenere premuto il tasto CH+, e verificate che il Boe-Bot avanzi.
- √ Ripetere il test con il tasto 1 key, dovrebbe avere lo stesso effetto.
- √ Provate i tasti CH+, 1, CH-, 2, VOL+, 3, VOL-, e 4.
- √ Divertitevi a guidare il Boe-Bot.

Per alcuni "Manovratori", il mio Boe-Bot sembra erratico, perchè? Ha a che vedere con il modo in cui il programma rileva lo start del messaggio IR. Non sempre lo rileva al primo colpo. Lo correggeremo nella prossima Attività.

```
' IR Remote for the Boe-Bot - 2BitRemoteBoeBot.bs2
' Control your Boe-Bot with an IR remote set to control a SONY TV
' with the 1-4 or CH+/- and VOL+/- keys.

'{$STAMP BS2}
'{$PBASIC 2.5}


time           VAR     Word(2)              ' SONY TV remote variables.
DEBUG "Press and hold a digit key (1-4) or CH+/- and VOL+/-..."

FREQOUT 4, 2000, 3000                       ' Start/reset indicator.

DO                                          ' Beginning of main loop.

  DO                                        ' Wait for rest between messages.
    PULSIN 9, 1, time(0)
  LOOP UNTIL time(0) > 1000

  PULSIN 9, 0, time(0)                      ' Measure/store data pulses.
  PULSIN 9, 0, time(1)
```

```
  ' Decide which maneuver to execute depending on the combination
  ' of pulse durations stored in the first two pulse measurements.

  IF (time(1) < 500) AND  (time(0) < 500) THEN
    PULSOUT 13, 850                         ' Forward
    PULSOUT 12, 650
  ELSEIF (time(1) < 500) AND (time(0) > 500) THEN
    PULSOUT 13, 650                         ' Backward
    PULSOUT 12, 850
  ELSEIF (time(1) > 500) AND (time(0) < 500) THEN
    PULSOUT 13, 850                         ' Right rotate
    PULSOUT 12, 850
  ELSEIF (time(1) > 500) AND (time(0) > 500) THEN
    PULSOUT 13, 650                         ' Left rotate
    PULSOUT 12, 650
  ENDIF

LOOP                                        ' Repeat main loop.
```

Il Vostro Turno – Spiegare perchè CH/VOL e 1-3 fanno la stessa cosa

- √ Tornate alla Table 1-1 a pagina 31 e confrontate le durate degli impulsi nelle righe **`time(1)`** e **`time(0)`** per 1 e CH+, 2 e CH-, 3 e VOL+, ed infine 4 e VOL-.
- √ Ora, spiegate perchè questi tasti fanno eseguire al Boe-Bot le stesse funzioni.
- √ Osservate la colonna **`time(4)`**, e spiegate come un TV SONY può distinguere tra i tasti CH+ e 1 keys.

ATTIVITÀ #5: AGGIUNGERE PRESTAZIONI AL BOE-BOT CON IR

2BitRemoteBoeBot.bs2 può essere espanso per eseguire più manovre. La Figura 1-16 mostra un disegno della tastiera del telecomando IR con le manovre già assegnate a ciascun tasto. Ottenere tutte queste funzioni complete è semplice, ma ci vuole un pò di lavoro extra per completare il compito. In attività successive, apprenderete un modo più universale di aggiungere o rimuovere tasti/pulsanti funzione con molta meno fatica.

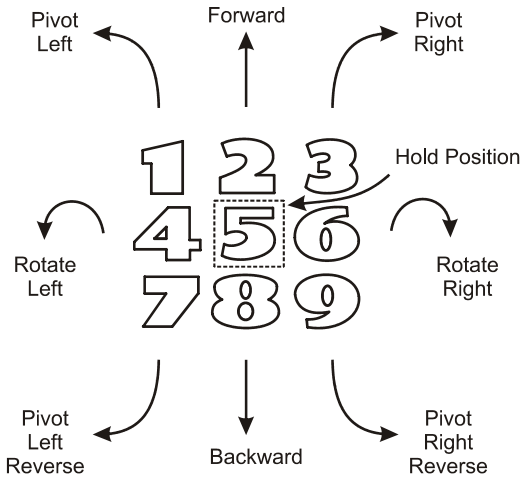


Figura 1-16
Controllo con la Tastiera Numerica.

Una Temporizzazione Cambia ed Aumentano gli IF…THEN

Ci sono due modifiche che dovrete fare a 2BitRemoteBoeBot.bs2 per fargli accettare questa specifica. Potreste già aver indovinato la prima modifica; il programma dovrà prendere la sua decisione con massimo tre impulsi forse quattro. Date ora un altra occhiata alla Table 1-1 a pagina 31. Se tutto quello che state usando per prendere le vostre decisioni sono **`time(1)`** e **`time(0)`**, il programma può facilmente confondere il tasto 5 con il tasto 1. similmente con i tasti 2 e 6, e così via. Per risolvere questo problema, potete usare il ragionamento **`IF…THEN`** basato su tre impulsi. Questo risolverebbe i tasti 1-8. il tasto 9 è ancora un problema, ma lo lasceremo per la sezione Il Vostro Turno.

Per misurare tre impulsi di dati, dovrete fare due modifiche a 2BitRemoteBoeBot.bs2:

- √ Modificare la dichiarazione della variabile **`time`** in modo che sia un array a tre word invece che a due.
- √ Aggiungere un terzo comando **`PULSIN`** per acquisire un terzo impulso e memorizzarlo nella variabile **`time(2)`**.
- √ Cambiare il prompt del Terminale di Deubug in "Press and hold a digit key (1-8)…"

Una volta che avete effettuato queste modifiche, ecco una dichiarazione **`IF…THEN`** che funzionerà con i tasti 1-8:

```
  IF (time(2) < 500) AND (time(1) < 500) AND (time(0) < 500) THEN
    PULSOUT 13, 750
    PULSOUT 12, 650
  ELSEIF (time(2) < 500) AND (time(1) < 500) AND (time(0) > 500) THEN
    PULSOUT 13, 850
    PULSOUT 12, 650
  ELSEIF (time(2) < 500) AND (time(1) > 500) AND (time(0) < 500) THEN
    PULSOUT 13, 850
    PULSOUT 12, 750
  ELSEIF (time(2) < 500) AND (time(1) > 500) AND (time(0) > 500) THEN
    PULSOUT 13, 650
    PULSOUT 12, 650
  ELSEIF (time(2) > 500) AND (time(1) < 500) AND (time(0) < 500) THEN
    PULSOUT 13, 750
    PULSOUT 12, 750
  ELSEIF (time(2) > 500) AND (time(1) < 500) AND (time(0) > 500) THEN
    PULSOUT 13, 850
    PULSOUT 12, 850
  ELSEIF (time(2) > 500) AND (time(1) > 500) AND (time(0) < 500) THEN
    PULSOUT 13, 750
    PULSOUT 12, 850
  ELSEIF (time(2) > 500) AND (time(1) > 500) AND (time(0) > 500) THEN
    PULSOUT 13, 650
    PULSOUT 12, 850
  ENDIF
```

Avere familiarità con il blocco di codice precedente tornerà utile per approfondire la tecnica di conteggio in binario, che studieremo nel prossimo Capitolo. Quest è un altra dichiarazione **IF…THEN** che potete usare e che esegue le stesse funzioni in modo più efficiente. Mentre le dichiarazioni **IF…THEN** con tre argomenti possono dover controllare il valore della variabile **time(2)** fino a otto volte, questa non controlla mai la variabile più di due volte.

```
IF (time(2) < 500) THEN
  IF time(1) < 500) AND (time(0) < 500) THEN
    PULSOUT 13, 750
    PULSOUT 12, 650
  ELSEIF (time(1) < 500) AND (time(0) > 500) THEN
      •
      •
      •
  ENDIF
ELSEIF (time(2) > 500) THEN
  IF (time(1) < 500) AND (time(0) < 500) THEN
    PULSOUT 13, 750
    PULSOUT 12, 750
  ELSEIF (time(1) < 500) AND (time(0) > 500) THEN
```

```
        •
        •
        •
  ENDIF
ENDIF
```

La prossima modifica che deve essere fatta è molto più subdola, e la ragione per il cambiamento non è necessariamente così ovvia. Per prima cosa, ecco la modifica che deve essere fatta:

√ Cambiare il comando **PULSIN** in un ciclo **DO…LOOP** che scansiona l'intervallo tra gli impulsi da questo:

```
DO
         PULSIN 9, 1, time(0)
LOOP UNTIL time(0) > 1000
```

Ad un comando **RCTIME**. Quando avete terminato le modifiche, dovrebbe essere simile a questo:

```
DO
  RCTIME 9, 1, time(0)
LOOP UNTIL time(0) > 1000
```

Assicuratevi che tutti i vostri programmi da questo momento in poi usino RCTIME invece di PULSIN per rilevare l'intervallo tra messaggi.

Questa è la ragione per cui questa modifica deve essere fatta. I comandi **PULSIN** combinati con le dichiarazioni **IF…THEN** per le decisioni possono impiegare più a lungo del tempo che impiega il messaggio IR a completarsi. Non è per il momento il caso, ma questo potrebbe cambiare man mano che aggiungete prestazioni. Il comando **PULSIN** deve rilevare sia il fronte di salita che di discesa all'inizio ed alla fine dell'intervallo mostrato nella Figura 1-17. Non appena i comandi **PULSIN** combinati con la parte decisionale eccedono il tempo di completamento del messaggio IR, il comando **PULSIN** perderà il fronte di salita dell'intervallo tra messaggi.

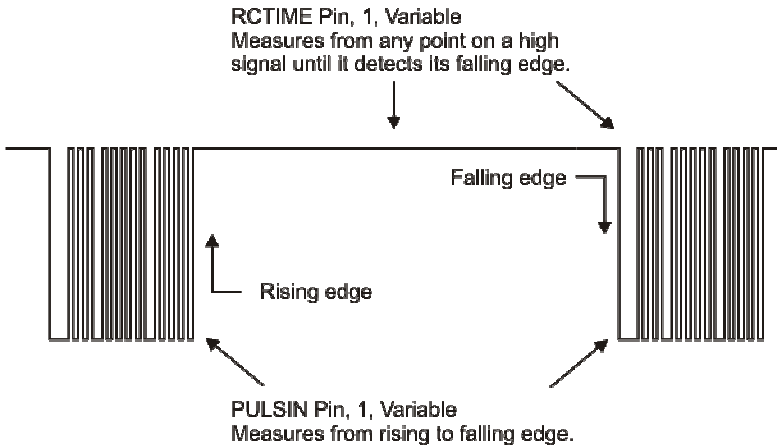


Figura 1-17 Confonto fra PULSIN e RCTIME per la Rivelazione dell' Intervallo tra Messaggi.

Se il comando **PULSIN** non rivela il fronte di salita dell'intervallo, il ciclo che cerca un impulso alto continuerà a cercare impulsi per trovarne uno maggiore di 1000. Eventualmente, magari circa 80 ms più tardi, il ciclo rivelerà l'intervallo dopo l'ennesima ripetizione del messaggio IR. Finalmente, a quel punto, il programma si muoverà ai comandi **IF…THEN** e **PULSOUT**. Questo ritardo causa un problema peer i servo. Se il tempo tra gli impulsi servo arriva fino a 90 ms a causa dell'incapacità del comando **PULSIN** di acquisire il primo intervallo tra i messaggi, i servo non funzioneranno correttamente o non funzioneranno affatto.

La soluzione di questo problema è nell'uso di **RCTIME**. Date una seconda occhiata alla Figura 1-17. Mentre **PULSIN** necessita sia del fronte di salita che del fronte di discesa per completare la sua misurazione, ad **RCTIME** serve solamente un fronte di discesa. Quindi, anche se il programma inizia a controllare l'intervallo tra messaggi solo dopo che è partito, sarà ancora in grado di riconoscerlo.

Programma Esempio 3BitRemoteBoeBot.bs2

Questo programma fà reagire il Boe-Bot quando si preme e si tiene premuto un dato tasto del telecomando. Le direzioni di ciascun tasto sono mostrate in Figura 1-16. Il tasto 9 non funziona correttamente, ma risolverete questo problema nella sezione Il Vostro Turno. È meglio assicurarsi che i tasti 1-8 funzionino prima di affrontare il tasto 9.

- √ Digitate e lanciate 3BitRemoteBoeBot.bs2.
- √ Testate i tasti e divertitevi per un pò a guidare il vostro Boe-Bot.
- √ Spiegate il problema del tasto 9 usando la Table 1-1 come riferimento.

```
' IR Remote for the Boe-Bot - 3BitRemoteBoeBot.bs2
' Control your Boe-Bot with keys 1-8 on a remote set to control a SONY TV.

'{$STAMP BS2}
'{$PBASIC 2.5}

time           VAR     Word(3)              ' SONY TV remote variables.

FREQOUT 4, 2000, 3000                       ' Start/reset indicator.
DEBUG "Press and hold a digit key (1-8)..."

DO                                          ' Beginning of main loop.

  DO                                        ' Wait for rest between messages.
    RCTIME 9, 1, time(0)
  LOOP UNTIL time(0) > 1000

  PULSIN 9, 0, time(0)                      ' Measure/store data pulses.
  PULSIN 9, 0, time(1)
  PULSIN 9, 0, time(2)

  ' Decide which maneuver to execute depending on the combination
  ' of pulse durations stored in the first three pulse measurements.

  IF (time(2) < 500) AND (time(1) < 500) AND (time(0) < 500) THEN
    PULSOUT 13, 750
    PULSOUT 12, 650
  ELSEIF (time(2) < 500) AND (time(1) < 500) AND (time(0) > 500) THEN
    PULSOUT 13, 850
    PULSOUT 12, 650
  ELSEIF (time(2) < 500) AND (time(1) > 500) AND (time(0) < 500) THEN
    PULSOUT 13, 850
    PULSOUT 12, 750
  ELSEIF (time(2) < 500) AND (time(1) > 500) AND (time(0) > 500) THEN
    PULSOUT 13, 650
    PULSOUT 12, 650
  ELSEIF (time(2) > 500) AND (time(1) < 500) AND (time(0) < 500) THEN
    PULSOUT 13, 750
    PULSOUT 12, 750
  ELSEIF (time(2) > 500) AND (time(1) < 500) AND (time(0) > 500) THEN
    PULSOUT 13, 850
    PULSOUT 12, 850
  ELSEIF (time(2) > 500) AND (time(1) > 500) AND (time(0) < 500) THEN
    PULSOUT 13, 750
    PULSOUT 12, 850
  ELSEIF (time(2) > 500) AND (time(1) > 500) AND (time(0) > 500) THEN
```

```
    PULSOUT 13, 650
    PULSOUT 12, 850
  ENDIF

LOOP                                          ' Repeat main loop.
```

Il Vostro Turno – Aggiustare il tasto 9

Se vi riferite alla Table 1-1 a pagina 31, noterete che il tasto 9 ha **time(3)** maggiore di 500 mentre il resto degli elementi dell'array **time** sono inferiori a 500. La ragione per cui il Boe-Bot ruota in avanti e verso sinistra è perchè le altre tre misure di **time** sono le stesse del tasto 1. per risolvere il problema, dovrete aumentare la dimensione dell'array **time** di una word ed aggiungere anche un altra misura con **PULSIN**. In ultimo, dovrete modificare la dichiarazione **IF…THEN** per rilevare il tasto 9 prima che venga scambiato per il tasto 1.

- √ Usate Save As dal menù File per salvare una copia di 3BitRemoteBoeBot.bs2 con il nome di 4BitRemoteBoeBot.bs2.
- √ Cambiate la dichiarazione della vostra variabile array **time** in modo che abbia quattro word chiamate **time** invece di tre.
- √ Aggiungete un comando **PULSIN** che misuri il quarto impulso di dati e lo memorizzi in **time(3)**.
- √ Espandete la dichiarazione **IF...THEN** aggiungendo una condizione all'inizio che controlli per vedere se il quarto impulsi di dati è maggiore di 500. se questo è il caso, allora il Boe-Bot dovrebbe ruotare all'indietro e verso destra. Potete eseguirlo sostituendo questa riga di codice:

  ```
  IF (time(2) < 500) AND (time(1) < 500) AND (time(0) < 500) THEN
  ```

  Con queste quattro righe:

  ```
  IF (time(3) > 500) THEN
    PULSOUT 13, 650
    PULSOUT 12, 750
  ELSEIF (time(2) < 500) AND (time(1) < 500) AND (time(0) < 500) THEN
  ```

- √ Lanciate e provate il programma, e cercate gli errori se è il caso.

SOMMARIO

Questo capitolo ha spiegato come il BASIC Stamp possa rilevare e memorizzare messaggi inviati da un telecomando universale. È stato presentato e spiegato il segnale specifico usato per il controllo delle TV SONY, quindi è stato misurato in dettaglio usando un circuito rivelatore ad infrarossi. La modulazione a larghezza di impulso (PWM) per la comunicazione è stata presentata come la base per la comunicazione tra il telecomando ed i componenti dei sistemi di intrattenimento. Le applicazioni comprendenti il controllo diretto a distanza del Boe-Bot hanno dimostrato come acquisire ed interpretare questi messaggi con l'aiuto dei comandi **`PULSIN`** e **`RCTIME`**, e degli array di variabili.

Domande

1. che cosa significa PWM, e quali sono i suoi usi?
2. Immaginate che il LED del telecomando IR stia lampeggiando a 38.5 kHz. Quindi, si ferma per un certo tempo prima di ricominciare. Quale segnale rileva il rivelatore IR durante questa pausa?
3. che cosa significa "programmare" un telecomando universale per inviare messaggi ad un TV SONY?
4. Quali sono i tre argomenti del comando **`PULSIN`** e cosa fà ciascuno dei tre?
5. come dovrete configurare un comando **`PULSIN`** per misurare un impulso positivo? Che cosa dovete fare per cambiare un comando **`PULSIN`** che misura impulsi positivi per fargli misurare impulsi negativi?
6. Ci sono diverse durate di impulsi bassi trasmesse dal telecomandi ad infrarossi, e ciascuna di esse ha il suo significato. In che modo potete programmare il BASIC Stamp per filtrare solo una durata particolare e scartare le altre?
7. Che cosa dura più a lungo, il messaggio del telecomando IR o l'intervallo tra messaggi? Nota: la vostra risposta può essere corretta per il controllo di TV SONY, ma può essere differente per altri protocolli.
8. che genere di variabile funziona meglio per memorizzare misure successive di durate di impulsi? Come dichiarate questa variabile?
9. In base alla durata degli impulsi come fattore per prendere decisioni, quali valori di soglia sono stati usati per operare delle scelte?

Esercizi

1. Dichiarare un array grande abbastanza per contenere una stringa di otto caratteri.
2. Spiegate quale sarebbe la differenza con la Figura 1-14 a pagina 27 se fosse stato premuto e mantenuto premuto il tasto 4 invece del tasto 5.

Progetti

1. Scrivete un programma che somma tutti gli impulsi IR e visualizza la durata di un messaggio IR in microsecond. Suggerimento: iniziate per prima cosa da RecordAndDisplayPwm.bs2. annotando e sommando tutti gli impulsi dei dati. Quindi filtrate, acquisite e sommate: gli intervalli tra messaggi, l'intervallo tra impulsi moltiplicati per dodici, e l'impulso di start. Dopo che avrete sommato gli impulsi dei dati, potete usare un elemento dell'array **`time`** per memorizzare il totale, ed altri tre per le altre misure.
2. Modificate 4BitRemoteBoeBot.bs2 in modo che faccia eseguire un cerchio completo ogni volta che viene premuto il tasto 5.
3. Modificate 4BitRemoteBoeBot.bs2 in modo che i tasti CH+/- e VOL+/- funzionino insieme agli altri tasti.

Soluzioni

Q1. PWM stà per (Pulse Width Modulation) modulazione a larghezza di impulso, e può essere usata per la conversione da digitale ad analogico, per il controllo di motori e per le comunicazioni.
Q2. Un segnale alto.
Q3. A pagina 6, si dice: " il termine "programmazione"si riferisce alla sequenza di tasti che verranno premuti sul telecomando per istruirlo ad emettere impulsi secondo il protocollo SONY ".
Q4. ***Pin*** seleziona il piedino I/O che controllerà l'impulso in arrivo, ***State*** configura il comando **PULSIN** per misurare impulsi sia positivi che negativi. ***Variable*** è una variabile che memorizza le misure delle durate degli impulsi in unità di 2 µs.
Q5. l' argomento ***State*** deve essere posto ad 1 per misurare impulsi positivi, mentre deve essere cambiato a 0 per misurare impulsi negativi.
Q6. Usate un ciclo per continuare a misurare gli impulsi, ma attivate un particolare blocco di codice solamente se la durata rientra in una certa gamma. Per questo scopo **IF…THEN** è funzionale. Per esempio, **IF (time > 975) AND (time < 1425) THEN** filtra gli impulsi maggiori di 1950 µs e minori di 2850 µs).
Q7. Gli intervalli durano più a lungo.
Q8. Un array di variabili è più funzionale per la memorizzazione di misure successive e correlate. Sommate il numero degli elementi all'argomento ***Size*** della dichiarazione di variabile (tra parentesi).
Q9. È conveniente usare un valore di 500 per distinguere tra impulsi che normalmente ricadono tra 300 (0.6 ms) e 600 (1.2 ms).

E1. **characters VAR Byte(8)**.
E2. l'elemento **time(2)** dell' array sarà probabilmente 364, mentre **time(1)** e **time(0)** saranno circa 660. Questo è stato ottenuto con l'aiuto della Table 1-1 a pagina 31.

P1. Iniziate con RecordAndDisplayPwm.bs2 ed usate vari elementi nell' array **time** per misurare gli elementi del messaggio: dodici impulsi di dati, l'intervallo tra i messaggi, dodici intervalli tra gli impulsi, l'impulso di start.

```
' IR Remote for the Boe-Bot - Capitolo1Project1.bs2
' Add all the pulse durations for total message time.

' {$STAMP BS2}
' {$PBASIC 2.5}
```

```
time           VAR     Word(12)         ' SONY TV remote variables
index          VAR     Nib

DO                                      ' Beginning of main loop

  DO                                    ' Wait for rest between messages
    PULSIN 9, 1, time(0)
  LOOP UNTIL time(0) > 1000

  PULSIN 9, 0, time(0)                  ' Measure/store data pulses
  PULSIN 9, 0, time(1)
  PULSIN 9, 0, time(2)
  PULSIN 9, 0, time(3)
  PULSIN 9, 0, time(4)
  PULSIN 9, 0, time(5)
  PULSIN 9, 0, time(6)
  PULSIN 9, 0, time(7)
  PULSIN 9, 0, time(8)
  PULSIN 9, 0, time(9)
  PULSIN 9, 0, time(10)
  PULSIN 9, 0, time(11)

  FOR index = 1 TO 11                   ' Add up data pulse durations.

    time(0) = time(0) + time(index)

  NEXT

  DO                                    ' Wait for rest between messages
    PULSIN 9, 1, time(1)
  LOOP UNTIL time(1) > 1000

  DO                                    ' Wait for start pulse
    PULSIN 9, 0, time(2)
  LOOP UNTIL (time(2) > 1000)

  DO                                    ' Wait for data pulse rest
    PULSIN 9, 1, time(3)
  LOOP UNTIL (time(3) > 1) AND (time(3) < 1000)

  time(3) = time(3) * 12                ' Twelve data pulse rests

  ' Add time(1) (rest between messages), time(2) (start pulse),
  ' time(3) (twelve rests between messages), and time(0) (the total
  ' data pulses).  This is the message total.

  time(0) = time(0) + time(1) + time(2) + time(3)
```

```
  DEBUG "time(0) = ",
        DEC time(0) * 2,
        " us", CR                    ' Display time in us

LOOP                                 ' Repeat main loop
```

P2. Iniziate con 4BitRemoteBoeBot.bs2.

√ Aggiungete questa dichiarazione di variabile:

```
counter  VAR Byte
```

√ Cancellate questi due comandi **PULSOUT**:

```
PULSOUT 13, 750
PULSOUT 12, 750
```

√ Sostituiteli con questo ciclo **FOR…NEXT**. ***EndValue*** che ha valore 76 Dovrà essere calibrato.

```
FOR counter = 1 TO 76
  PULSOUT 13, 650
  PULSOUT 12, 650
  PAUSE 20
NEXT
```

P3. Iniziate con 4BitRemoteBoeBot.bs2.

√ Aumentate l'array **time** a 5 elementi.

√ Aggiungete un quinto comando **PULSIN** per caricare **time(4).**

√ Cambiate la dichiarazione **IF** in **ELSEIF**.

√ Prima della dichiarazione **ELSEIF** (che avete appena sostituito ad **IF**) aggiungete questo codice:

```
IF (time(4) > 500) THEN
  IF (time(1) < 500) AND  (time(0) < 500) THEN
    PULSOUT 13, 850
    PULSOUT 12, 650
  ELSEIF (time(1) < 500) AND (time(0) > 500) THEN
    PULSOUT 13, 650
    PULSOUT 12, 850
  ELSEIF (time(1) > 500) AND (time(0) < 500) THEN
    PULSOUT 13, 850
    PULSOUT 12, 850
  ELSEIF (time(1) > 500) AND (time(0) > 500) THEN
    PULSOUT 13, 650
    PULSOUT 12, 650
  ENDIF
```

Capitolo 2: Creare ed Usare Applicazioni Remote

PROGRAMMI RIUSABILI

In questo capitolo, svilupperete e collauderete due diversi programmi applicativi per telecomandi IR. Li proverete da soli ed anche con nuove applicazioni per il Boe-Bot. Essi renderanno possibile con meno lavoro la realizzazione tutti i generi di programmi telecomandati per il Boe-Bot. questi programmi applicativi possono anche essere usati con molti altri progetti. Pensate a tutte le macchine ed alle invenzioni che hanno delle tastiere. Potrete progettare la vostra versione di queste invenzioni con un rivelatore di infrarossi ed un telecomando universale.

ATTIVITÀ #1: INTERPRETARE I MESSAGGI IR

La chiave per utilizzare le misure degli impulsi che avete raccolto nel capitolo precedente è la decodifica. Nel caso dei messaggi che il telecomando universale invia, decodificarli significa convertire una serie di misura delle durate di impulsi in un singolo valore che il vostro programma PBASIC possa usare per prendere decisioni.

Decodificare

a: convertire (come nel caso di un messaggio codificato) in una forma intelleggibile
b: riconoscere ed interpretare (un segnale elettronico)

Fonte: Dizionario OnLine Merriam-Webster – www.merriam-webster.com

In questa attività, scriverete proggrammi che decodificano i messaggi PWM inviati dal telecomando infrarosso. Indagherete anche le relazioni tra questi valori decodificati ed i tasti sul telecomando. Se esiste una relazione, ciò faciliterà la realizzazione dei programmi per telecomando. Se è più facile scrivere un programma per un' applicazione remota, sarete in grado di scrivete con meno lavoro, applicazioni per il Boe-Bot più potenti.

Contare in Binario e Contare in Decimale

Ecco come si conta fino a 20 con i numeri binari:

Binario	0	1	10	11	100	101	110	111	1000	1001	1010	1011
Decimale	0	1	2	3	4	5	6	7	8	9	10	11

Binario	1100	1101	1110	1111	10000	10001	10010	10011	10100
Decimale	12	13	14	15	16	17	18	19	20

Notare che ci vogliono, per contare fino a 15 quattro bit o meno, mentre per contare fino a 20 sia necessario un quinto bit.

Nella numerazione binaria, le cifre vengono normalmente chiamate bit. Laddove una cifra può essere da 0 a 9 nella nnumerazione decimale, un bit, nella numerazione binaria può solamente essere 1 o 0. La Figura 2-1 mostra uno schema dei bit di un numero binario ad otto bit (byte). Il bit più a destra è bit-0, il successivo è bit-1, quindi bit-2, fino a bit-7.

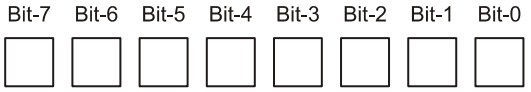


Figura 2-1
I Bit in un Byte Binario

Ciascuna casella può contenere 1 o 0.

Ciascun bit in un numero binario, indica se una certa potenza di 2 è contenuta o no. Bit-0 indica se il valore 1 è presente o no nel numero binario, bit-1 indica se il valore 2 è presente o meno, bit-2 indica se 4 è presente e così via. Anche se avete un numero binario molto grande potete sempre conoscere il valore di un singolo bit perchè è 2^{bit}.

Questo è un esempio di come prendere 2 ed elevarlo alla potenza della sua posizione di bit. Bit-0 indica che il valore è 1 perchè $2^0 = 1$. Similmente, bit-1 indica che 2 è presente perchè $2^1 = 2$. Bit-2 indica che 4 è presente perchè $2^2 = 4$. Bit-6 indica che 64 è presente perchè $2^6 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 64$.

La Figura 2-2 mostra come convertire un numero binario fino ad otto bit (byte) nel suo corrispondente valore decimale.

- Per prima cosa, inserite il numero il numero binario nelle caselle della riga superiore. Assicuratevi di inserire il bit più a destra nella casella più a destra, quindi continuate nella casella successiva verso sinistra fino ad inserire tutti ed otto i bit nelle caselle.

- Secondo, moltiplicate ciascun bit per la sua potenza di due, ed inserite il risultato nella casella immediatamente sottostante.
- Terzo, sommare tutti i prodotti ed inserite il risultato a destra del segno =.

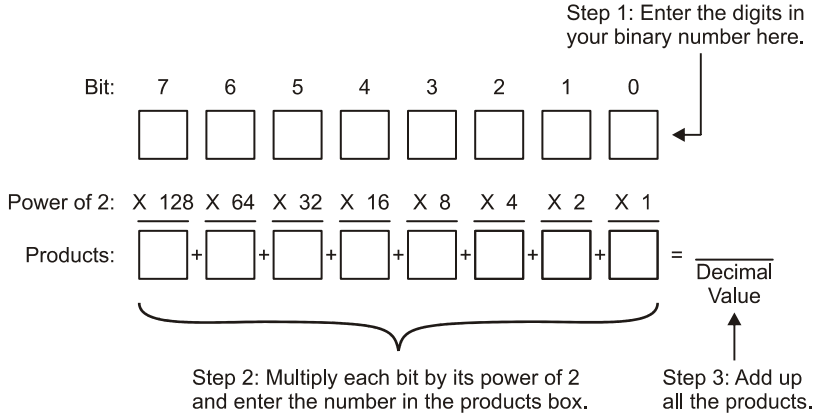


Figura 2-2 Conversione da Binario a Decimale

Nella Figura 2-3 seguite questi passi per calcolare il valore decimale del numero binario 10011.

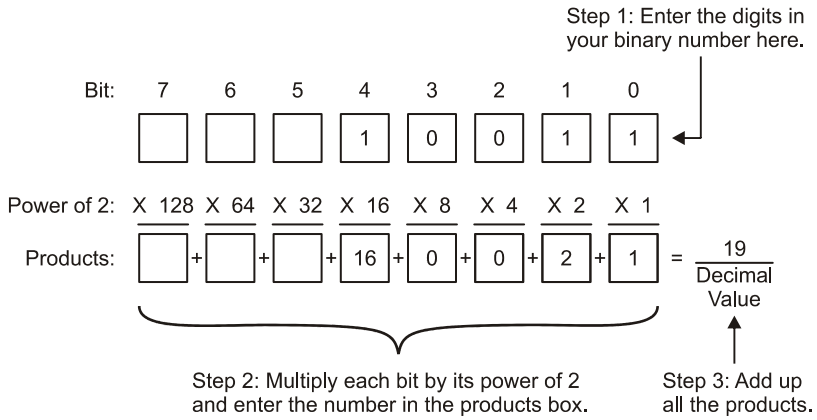


Figura 2-3 Calcolare l'equivalente Decimale del numero Binario10011

Programma Esempio – BinaryToDecimal.bs2

Sebbene possiate scrivere un programma PBASIC che esegua ciascuno dei passi appena studiati, i modificatori **`BIN`** e **`DEC`** possono facilitare il compito.

- √ Digitate e lanciate BinaryToDecimal.bs2.
- √ Clickate nella finestra di trasmissione del Terminale di Debug.
- √ Digitate fino ad otto cifre binarie (uno e zero), quindi premete il tasto Enter.
- √ Calcolate a mano il valore usando la procedura mostrata in Figura 2-2.
- √ Confrontate il risultato del vostro calcolo manuale con quello mostato nel Terminale di Debug.

```
' IR Remote for the Boe-Bot - BinaryToDecimal.bs2
' Enter a binary value into the Debug Terminal's Transmit Windowpane,
' and get the decimal value in the Receive Windowpane.

'{$STAMP BS2}
'{$PBASIC 2.5}

value          VAR     Byte

DO

  DEBUG "Enter binary value: "
  DEBUGIN BIN value
  DEBUG "Decimal value is: ", DEC value, CR

LOOP
```

Il Vostro Turno – Contare in Binario

Inserendo ciascuno dei numeri binari elencati all'inizio di pagina 44, vi impratichirete nel conteggio binario. Specialmente se approfondirete ulteriormente i mocrocontrollori, la capacità di contare in binario, sarà una capacità su cui farete affidamento in continuazione.

- √ Mentre il programma è in funzione, inserite nella finestra di trasmissione del Terminale di Debug, ciascuno dei ventuno valori binari elencati all'inizio di pagina 44.
- √ Assicuratevi che la conversione decimale verifichi che stiate contando correttamente.

Abilitare e Disabilitare Bit con il Modificatore .BIT

Sebbene il modificatore **BIN** faciliti l'inserimento dei numeri binari nel Terminale di Debug, non è molto utile per la conversione di una serie di impulsi in un numero binario. Ciascun impulso corrisponde ad un bit differente nel numero binario che il telecomando stà trasmettendo. Questo significa che ciascun impulso deve essere tradotto in 1 o 0, e quindi il bit nella posizione corrispondente deve essere abilitato o disabilitato.

Il modificatore di variabile **.BIT** può essere usato per abilitare o disabilitare bit in una variabile. Ciò rende possibile selezionare singoli bit in una data variabile. Diciamo di avere una variabile byte chiamata **value**, e di voler disabilitare il bit-5 ed abilitare il bit-6. Questo è il modo di farlo con il modificatore di variabile **.BIT**:

```
value.BIT5 = 0
value.BIT6 = 1
```

Il prossimo programma esempio spiega come usare il modificatore **.BIT** per abilitare e disabilitare bit in una variabile byte con l'aiuto delle finestre di trasmissione e ricezione del Terminale di Debug. Esso usa tre variabili, un byte chiamato **value**, un nibble chiamato **index,** ed un bit chiamato **setClear**. Memorizzando valori in **index** ed in **setClear** con il comando **DEBUGIN**, potrete scegliere qualsiasi bit nella variabile **value** ed abilitarlo ad 1 o disabilitarlo a 0 con questi comandi:

```
DEBUGIN DEC1 index

DEBUGIN BIN1 setClear

IF index = 0 THEN value.BIT0 = setClear
IF index = 1 THEN value.BIT1 = setClear
IF index = 2 THEN value.BIT2 = setClear
IF index = 3 THEN value.BIT3 = setClear
IF index = 4 THEN value.BIT4 = setClear
IF index = 5 THEN value.BIT5 = setClear
IF index = 6 THEN value.BIT6 = setClear
IF index = 7 THEN value.BIT7 = setClear
```

Consigli di Codifica – il modificatore `.LOWBIT`: Il modificatore **`.LOWBIT`** può essere usato per trattare i bit di una variabile come elementi di un array. Ecco come potete usare il modificatore **`.LOWBIT`** per sostituire le otto dichiarazioni **`IF…THEN`** di cui abbiamo appena parlato.

```
value.LOWBIT(index) = setClear
```

Questa tecnica non può essere usata con gli esempi di comandi remoti IR in arrivo perchè richiederebbe una calibrazione più sofisticata con le misurazioni degli impulsi IR remoti. Ma anche così, è un esercizio utile, sostituire le otto dichiarazioni **`IF…THEN`** con questo singolo comando nel prossimo programma esempio. Verificate che ambedue le operazioni effettuino la stessa operazione.

Programma Esempio – SetAndClearWithDotBit.bs2

La Figura 2-4 smostra un esempio di cosa potete fare con SetAndClearWithDotBit.bs2. digitando una cifra nella finestra di trasmissione quando si è invitati a farlo per il "bit index" potete selezionare il bit che volete cambiare nella variabile **`value`**. Quindi, digitando 1 o 0 nella finestra di trasmissionequando mostra "1 to set or 0 to clear", potete controllare se il bit in **`value`** viene abilitato 1 o disabilitato 0.

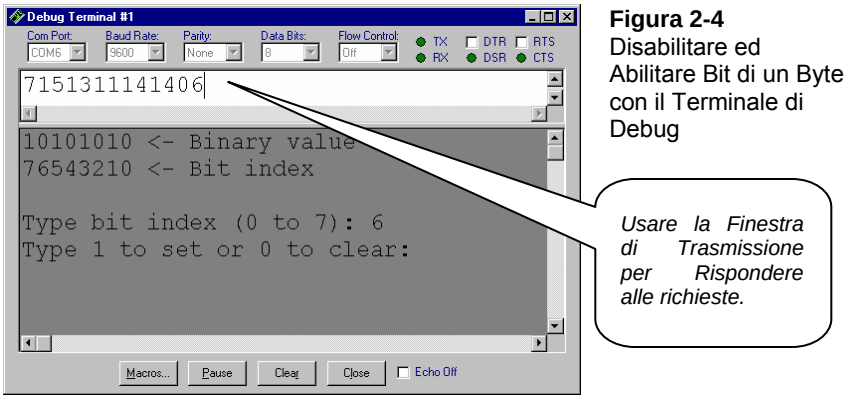


Figura 2-4
Disabilitare ed Abilitare Bit di un Byte con il Terminale di Debug

- √ Digitate, Salvate e Lanciate SetAndClearWithDotBit.bs2.
- √ Quando apparirà la richiesta "Type bit index (0 to 7): ", decidete quale bit volete cambiare, e digitate quella cifra.
- √ Quando apparirà la richiesta "Type 1 to set or 0 to clear: ", digitate i tasti 1 o 0.

Il programa farà una pausa di mezzo secondo, quindi azzererà il display.

- √ Contollate ed assicuratevi che i cambiamenti che avete operato appaiano in "Binary value".
- √ Sperimentate con l'abilitazione e la disabilitazione nel "Binary value".

```
' IR Remote for the Boe-Bot - SetAndClearWithDotBit.bs2
' Use the Debug Terminal's Transmit Windowpane to choose a bit in
' the value variable and set or clear it.

'{$STAMP BS2}
'{$PBASIC 2.5}

value          VAR     Byte
index          VAR     Nib
setClear       VAR     Bit

DO

  DEBUG CLS,
        BIN8 value, " <- Binary value", CR,
        "76543210 <- Bit index", CR, CR,
        "Type bit index (0 to 7): "

  DEBUGIN DEC1 index

  DEBUG CR, "Type 1 to set or 0 to clear: "

  DEBUGIN BIN1 setClear

  IF index = 0 THEN value.BIT0 = setClear
  IF index = 1 THEN value.BIT1 = setClear
  IF index = 2 THEN value.BIT2 = setClear
  IF index = 3 THEN value.BIT3 = setClear
  IF index = 4 THEN value.BIT4 = setClear
  IF index = 5 THEN value.BIT5 = setClear
  IF index = 6 THEN value.BIT6 = setClear
  IF index = 7 THEN value.BIT7 = setClear

  PAUSE 500

LOOP
```

Come Funziona SetAndClearWithDotBit.bs2

La variabile **`value`** è il byte, con 8 bit che voi potete porre a 0 o ad 1 con il Terminale di Debug. La variabile **`index`** memorizza il valore che determina quale bit in **`value`** verrà abilitato/disabilitato, e la variabile **`setClear`** memorizza un 1 o uno 0.

```
value          VAR     Byte
index          VAR     Nib
setClear       VAR     Bit
```

I rimanenti comandi sono annidati nel ciclo **DO…LOOP main**.

Un comando **DEBUG CLS** cancella la finestra di ricezione, visualizza la rappresentazione binaria ad 8 bit della variabile **value**, quindi visualizza il valore indice di ciascun bit, seguito dalla prima richiesta per l'utente (voi) di inserire un numero.

```
DEBUG CLS,
      BIN8 value, " <- Binary value", CR,
      "76543210 <- Bit index", CR, CR,
      "Type bit index (0 to 7): "
```

Il valore di **index** viene impostato con un comando **DEBUGIN** seguito da un secondo comando **DEBUG**, che visualizza una seconda richiesta. Quindi un secondo comando **DEBUGIN** preleva il valore di **setClear** dalla finestra di trasmissione del Terminale di Debug.

```
DEBUGIN DEC1 index

DEBUG CR, "Type 1 to set or 0 to clear: "

 DEBUGIN BIN1 setClear
```

Questo blocco di codice memorizza il valore di **setClear** nel bit che avete scelto nella variabile **value**. La serie di dichiarazioni **IF…THEN** esamina la variabile **index**. Quando viene trovata una corrispondenza, al bit corrispondente di **value** viene assegnato l' 1 o lo 0 che era stato memorizzato in **setClear**.

```
IF index = 0 THEN value.BIT0 = setClear
IF index = 1 THEN value.BIT1 = setClear
IF index = 2 THEN value.BIT2 = setClear
IF index = 3 THEN value.BIT3 = setClear
IF index = 4 THEN value.BIT4 = setClear
IF index = 5 THEN value.BIT5 = setClear
IF index = 6 THEN value.BIT6 = setClear
IF index = 7 THEN value.BIT7 = setClear
```

Dopodichè, **PAUSE 500** attende per mezzo secondo, quindi il ciclo **DO…LOOP** main si ripete. Quando lo fà, lo schermo viene cancellato, e viene visualizzata la nuova rappresentazione della variabile **value**.

Il Vostro Turno – Aggiungere la Conversione Decimale al Programma

Questo programma può essere anche un utile strumento per esaminare la relazione tra numeri binari e decimali.

- √ Salvate SetAndClearWithDotBit.bs2 in SetAndClearWithDotBitYourTurn.bs2.
- √ Modificate il comando **DEBUG** in modo che sia:

```
DEBUG CLS,
      BIN8 value, " <- Binary value", CR,
      "76543210 <- Bit index", CR,
      "Decimal value: ", DEC3 value, CR, CR,
      "Type bit index (0 to 7): ", CR
```

- √ Lanciate la vostra versione modificata del programma. Dovrebbe ora visualizzare l'equivalente decimale del numero binario i cui bit voi state abilitando o disabilitando.
- √ Partendo con il bit più a destra (bit-0), abilitatelo, annotate il valore e disabilitatelo.
- √ Ripetere con bit-1, quindi con bit-2, su fino a bit-7.
- √ Spiegate che cosa avete osservato.
- √ Quindi provate ad abilitare e disabilitare i bit in modo che possiate contare di nuovo da 0 a 20 in binario.

Convertire le Durate degli Impulsi IR in Valori Decimali

Useremo una variabile chiamata **remoteCode** per memorizzare il valore trasmesso dagli impulsi IR dei messaggi. Dal momento che il programma abiliterà i bit solamente per un dato impulso superiore ad 1 ms, tutti i bit in **remoteCode** dovranno essere disabilitati prima che le misure degli impulsi vengano acquisite. Il modo più semplice per disabilitare tutti i bit in una variabile è semplicemente impostare il suo valore uguale a zero. Se una variabile byte memorizza il vlore 0, in effetti stà memorizzando il numero binario 00000000. il chè significa che tutti i bit sono disabilitati.

```
remoteCode = 0
```

Quindi, attenderemo per l'itervallo tra i messaggi IR con un ciclo **DO…LOOP** che avevamo studiato nel Capitolo 1, Attività 5.

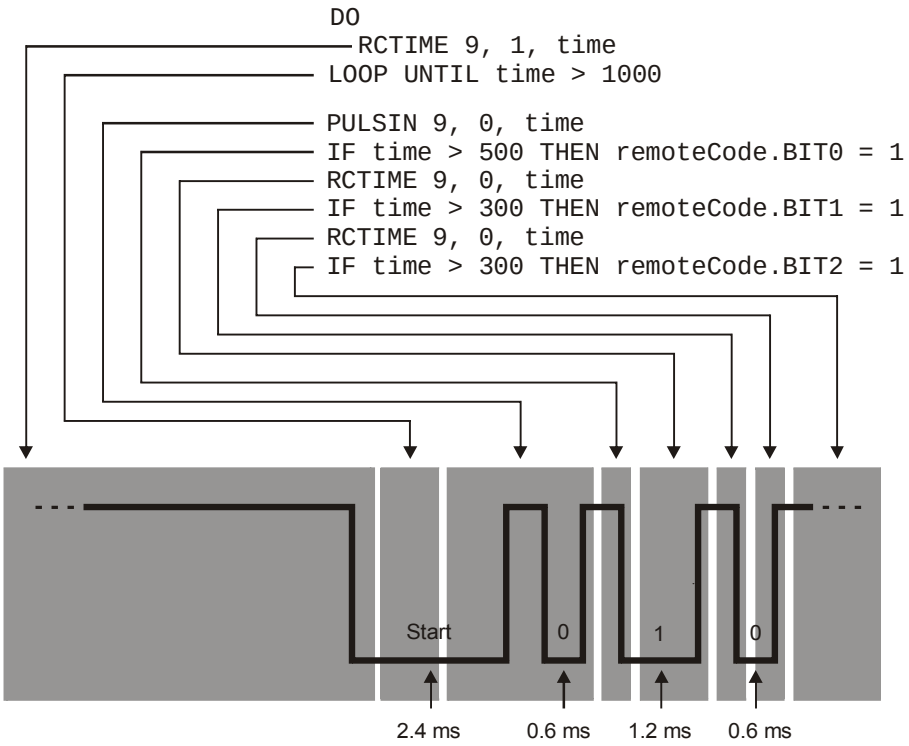
```
DO
  RCTIME 9, 1, time
LOOP UNTIL time > 1000
```

Quindi, inizieremo a misurare gli impulsi dei dati. Dopo che ciascun impulso dati è stato misurato, si controlla per vedere se è durato sufficientemente per significare un 1 binario. Se questo è il caso, abilitare il bit corrispondente nella variabile **remoteCode**; altrimenti viene lasciato disabilitato

```
PULSIN 9, 0, time                       ' Measure pulse.
IF time > 500 THEN remoteCode.BIT0 = 1  ' Set (or leave clear) bit-0.
RCTIME 9, 0, time                       ' Measure next pulse.
IF time > 300 THEN remoteCode.BIT1 = 1  ' Set (or leave clear) bit-1.
RCTIME 9, 0, time                       ' etc.
IF time > 300 THEN remoteCode.BIT2 = 1
RCTIME 9, 0, time
IF time > 300 THEN remoteCode.BIT3 = 1
RCTIME 9, 0, time
IF time > 300 THEN remoteCode.BIT4 = 1
RCTIME 9, 0, time
IF time > 300 THEN remoteCode.BIT5 = 1
RCTIME 9, 0, time
IF time > 300 THEN remoteCode.BIT6 = 1
```

Perchè usare **PULSIN** per misurare il primo impulso ed **RCTIME** per misurare il resto? Risparmiare sei variabili word vale un pò di ginnastica con il codice. Questa routine usa solamente una variabile **time**, non un array di sette come negli esempi del capitolo 1. La Figura 2-5 mostra la temporizzazione dell'esecuzione dei comandi in relazione agli impulsi IR dei messaggi in arrivo. Da sinistra a desta, c'è abbastanza tempo per il ciclo **DO…LOOP** di rilevare quando un impulso positivo è maggiore di 1000 perchè l'impulso di start che segue dura 2.4 ms. C'è anche tempo sufficiente per eseguire un comando **PULSIN** per acquisire il primo impulso dati. Il problema si verifica quando viene eseguita la prima dichiarazione **IF…THEN**. Ricordate che stiamo cercando di usare solamente una variabile **time** per acquisire le misure di tutti gli impulsi, non un array di sette variabili. A causa del fatto che la variabile **time** deve essere riusata per misurare il prossimo impulso, il valore corrente deve essere esaminato per abilitare o disabilitare **remoteCode.BIT0**.

Figura 2-5: Temporizzazione dell'Esecuzione dei Comandi in Relazione agli impulsi IR in arrivo



La ragione per cui per misurare il prossimo impulso deve essere usato **RCTIME** è perchè la dichiarazione **IF…THEN** non termina che dopo l'inizio del prossimo impulso di dati. Dal momento che l'impulso di start (il suo fronte negativo) è stato perso mentre veniva eseguita la dichiarazione **IF…THEN** statement, **PULSIN** non può rilevare correttamente l'inizio dell'impulso. Invece, **PULSIN** perderebbe quell'impulso dati e misurerebbe il successivo, rendendo possibile solamente l'acquisizione di metà degli impulsi dati di un messaggio IR. D'altra parte, può essere usato **RCTIME** per misurare ciò che rimane dell'impulso dati, e per misurare anche tutti gli altri impulsi dati rimanenti.

Tramite esperimenti simili a quelli del Capitolo 1, Attività #2, le misure successive di **RCTIME** memorizza in **time** valori intorno a 450 per un impulso 1 binario o 150 per un

impulso 0 binario. 300 è a mezza strada tra questi due valori, e viene usato per decidere se il resto delle misure degli impulsi sono 1 binario o 0 binario. Dichiarazioni **`IF…THEN`** seguono ciascuna di queste misure degli impulsi per abilitare o disabilitare da **`remoteCode.BIT1`** a **`remoteCode.BIT6`** basandosi su questo valore di 300.

Programma Esempio: PulsesToByteValue.bs2

Questo programma illustra i concetti che sono stati spiegati fino ad ora, decodificando il messaggio PWM del telecomando e visualizzandolo come valore decimale. Ogni volta che vorrete sapere qual'è il codice per uno dei tasti del telecomando, potrete usare questo programma. Quando conoscete qual'è il codice di ogni tasto del telecomando, scrivere programmi che fanno agire il Boe-Bot in base ai messaggi PWM ricevuti usando **`IF…THEN`** e **`SELECT…CASE`** diviene molto più facile.

- √ Digitate e lanciate PulsesToByteValue.bs2.
- √ Premere i vari tasti del telecomando e trovate qual'è il valore decimale di ciascuno.
- √ Testate i tasti da 1 a 9 e spiegate la relazione esistente tra il valore decimale visualizzato nel Terminale di Debug ed il valore del numero della tastiera.
- √ Provate il tasto 0. il valore del tasto zero presenta un problema di programmazione?
- √ Confrontate la versione binaria della variabile **`remoteCode`** visualizzata nel Terminale di Debug con le vostre annotazioni nella Table 1-1 a pagina 31. se un certo bit nella variabile **`remoteCode`** è 1, la misura corrispondente di **`time(index)`** nella Table 1-1 dovrebbe essere maggiore di 500. se il bit in **`remoteCode`** è 0, il valore corrispondente nella misura **`time(index)`** dovrebbe essere minore di 500.

```
' IR Remote for the Boe-Bot - PulsesToByteValue.bs2
' Display the binary and decimal values of the lower seven bits of
' IR message.

'{$STAMP BS2}
'{$PBASIC 2.5}

time           VAR     Word                  ' SONY TV remote variables.
remoteCode     VAR     Byte

DEBUG "Binary Value  Decimal Value", CR,     ' Display heading.
      "Bit 76543210              ", CR,
      "------------  -------------"
```

```
DO                                          ' Beginning of main loop.

  remoteCode = 0                            ' Clear all bits in remoteCode.

  DO                                        ' Wait for rest between messages.
    RCTIME 9, 1, time
  LOOP UNTIL time > 1000

  PULSIN 9, 0, time                         ' Measure pulse.
  IF time > 500 THEN remoteCode.BIT0 = 1    ' Set (or leave clear) bit-0.
  RCTIME 9, 0, time                         ' Measure next pulse.
  IF time > 300 THEN remoteCode.BIT1 = 1    ' Set (or leave clear) bit-1.
  RCTIME 9, 0, time                         ' etc.
  IF time > 300 THEN remoteCode.BIT2 = 1
  RCTIME 9, 0, time
  IF time > 300 THEN remoteCode.BIT3 = 1
  RCTIME 9, 0, time
  IF time > 300 THEN remoteCode.BIT4 = 1
  RCTIME 9, 0, time
  IF time > 300 THEN remoteCode.BIT5 = 1
  RCTIME 9, 0, time
  IF time > 300 THEN remoteCode.BIT6 = 1

  DEBUG CRSRXY, 4, 3, BIN8 remoteCode,      ' Display keypad code.
        CRSRXY, 14, 3, DEC2 remoteCode


LOOP                                        ' Repeat main loop.
```

Come Funziona PulsesToByteValue.bs2

Per questa applicazione, non ci interessa se il telecomando stà inviando messaggi ad un TV, VCR, o Decoder, quindi il bit più significativo nel messaggio non ci interessa. Questo programma acquisisce sette impulsi perchè questo è tutto ciò che è necessario per acquisire i tasti del telecomando TV. Inoltre sette misure impiegano meno tempo per essere eseguite, lasciando al BASIC Stamp altro tempo per prendere altre misure.

Ricordate che un impulso di circa 1.2 ms è un binario-1, mentre un impulso di circa 0.6 ms è un binario-0. per il primo comando **`PULSIN`**, se l'elemento dell'array **`time`** memorizza un valore maggiore di 500 (1000 µs = 1 ms), deve essere un impulso 1 binario. Similmente, se l'elemento dell'array **`time`** memorizza un valore minore di 500, deve essere un impulso 0 binario. Per i comandi **`RCTIME`** che seguono, se **`time`** memorizza un valore maggiore di 300, deve essere un 1 binario; altrimenti, è uno 0 binario.

Il Vostro Turno – Correzione dei Valori della Tastiera

Le regole secondo cui valori numerici corrispondono ai tasti possono confondere. Quando **remoteCode** è 0, il tasto premuto in realtà è l' 1. Quando **remoteCode** è 1, il tasto premuto in realtà è 2, e così via fino a che **remoteCode** è 8, che significa che il tasto 9 è stato premuto. Ma ora, quando **remoteCode** memorizza 9, significa che è stato premuto il tasto 0!

Potete aggiustare questo problema con un paio di dichiarazioni **IF…THEN**. Queste dichiarazioni **IF…THEN** possono regolare il valore memorizzato nella variabile **remoteCode** in modo tale che corrisponda al tasto premuto sulla tastiera. In altre parole, quando premete 5, **remoteCode** memorizza 5. quando premete 8, **remoteCode** memorizza 8. e più importante, quando premete 0, **remoteCode** memorizza 0.

- √ Salvate PulsesToByteValue.bs2 con il nome PulsesToByteValueYourTurn.bs2.
- √ Modificate il programma inserendo questi due comandi tra l'ultima dichiarazione **IF...THEN** ed il comando **DEBUG**.

```
IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
IF (remoteCode = 10) THEN remoteCode = 0
```

- √ Lanciate il programma e verificate che il valore memorizzato da **remoteCode** ora corrisponda al tasto premuto sulla tastiera del telecomando IR.
- √ Scrivete un breve rapporto spiegando in che modo questi due comandi ottengono lo scopo.
- √ Aggiornate la Table 2-1 seguente, e marcate questa pagina peer riferimento su come i valori numerici corrispondono ai tasti premuti sulla tastiera.

Table 2-1: Codici dei Tasti del Telecomando	
Tasto	**Valore Decimale**
0-9	
VOL-	
VOL+	
CH-	
CH+	
ENTER	
POWER	

ATTIVITÀ #2: PROGETTARE UN PROGRAMMA REMOTO RIUSABILE

Fino a questo punto, avete completato un programma che effettua l'acquisizione e la decodifica di messaggi IR remoti da un telecomando per segnali TV SONY. Prima di muoverci ad altre applicazioni per il Boe-Bot, è meglio riscrivere il programma in modo che funzioni dall'interno di subroutine. Insieme alle subroutine, il programma dovrebbe anche avere direttive **CON** per i tasti non numerici e dichiarazioni **VAR** per le variabili usate nelle subroutine.

Assemblaggio di una Applicazione per la Lettura del Telecomando IR

Il prossimo programma esempio è una versione più generale e riusabile del programma PulsesToByteValueYourTurn.bs2. queste sono le variazioni che gli sono state fatte:

- Costanti – valori di tasti con nomi significativi
  La Table 2-1 è stata usata per assemblare un elenco di direttive **CON**.
- Variabili – le variabili che devono essere usate con le subroutine
  Le dichiarazioni **VAR** *di PulsesToByteValue.bs2 sono state date nelle rispettive sezioni.*
- Routine Principale – chiama le subroutine e visualizza i dati
  È stata inserita una routine principale molto semplice che visualizza i codici del telecomando IR nel Terminale di Debug insieme con un commento per aggiungere il vostro codice.

- Subroutine – acquisisce gli impulsi dei messaggi dal Telecomando IR e decodifica

  Questa dovrebbe contenere tutto ciò che c'è all'interno del ciclo **DO**...**LOOP** *principale di PulsesToByteValueYourTurn.bs2.*

Per provare il telecomando, verrà usata la dichiarazione **SELECT…CASE** per assicurarsi che il programma riconosca tutti i diversi tasti. Ricordate, **SELECT…CASE** vi permette di selezionare una variabile, valutarla caso per caso. Potete usare singoli valori, un elenco di valori separati da virgole o una gamma di valori.

Programma Esempio: IrRemoteButtons.bs2

√ Digitate, salvate e lanciate IrRemoteButtons.bs2.
√ Premere e rilasciare ciascun tasto sul telecomando e verificate che venga visualizzata la cifra corretta.
√ Provate i tasti ENTER, CH+, CH-, VOL+, VOL-, e POWER. Verificare che i valori visualizzati coincidano con le direttive **CON** del programma.

Vi converrà salvare molte copie di questo file, per poi modificare queste copie per effettuare una varietà di funzioni diverse con il Boe-Bot.

√ Assicuratevi di salvare questo programma con il nome IrRemoteButtons.bs2.

Questo è un esempio di slavataggio e modifica di una copia di IrRemoteButtons.bs2.

√ Usate il comando File → Save As per salvare una copia di questo programma. Usate il nome TestIrRemoteButtons.bs2.
√ Sostituite la routine main esistente con questo blocco di codice:

```
DO

  GOSUB Get_Ir_Remote_Code

  DEBUG CLS, "Remote code: "

  SELECT remoteCode
    CASE 0 TO 9
      DEBUG DEC remoteCode
    CASE Enter
      DEBUG "ENTER"
    CASE ChUp
      DEBUG "CH+"
```

```
      CASE ChDn
        DEBUG "CH-"
      CASE VolUp
        DEBUG "VOL+"
      CASE VolDn
        DEBUG "VOL-"
      CASE Power
        DEBUG "POWER"
      CASE ELSE
        DEBUG DEC remoteCode, " (unrecognized)"
    ENDSELECT

    DEBUG CLREOL

  LOOP
```

- √ Lanciate TestIrRemoteButtons.bs2 e provate i tasti del telecomando così come gli altri tasti elencati nella sezione dichiarazione delle costanti.
- √ Provate i tasti SLEEP, MUTE, e LAST, che cosa succede? La porzione Il Vostro Turno di questa attività vi darà qualche suggerimento per la soluzione di questo problema.

```
' -----[ Title ]-----------------------------------------------------------
' IR Remote for the Boe-Bot - IrRemoteButtons.bs2
' Capture and store button codes sent by a universal remote conFigurad to
' control a SONY TV.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ I/O Definitions ]--------------------------------------------------

' SONY TV IR remote declaration - input received from IR detector

IrDet          PIN     9

' -----[ Constants ]-------------------------------------------------------

' SONY TV IR remote constants for non-keypad buttons

Enter          CON     11
ChUp           CON     16
ChDn           CON     17
VolUp          CON     18
VolDn          CON     19
Power          CON     21

' -----[ Variables ]-------------------------------------------------------
```

```
' SONY TV IR remote variables

irPulse        VAR     Word
remoteCode     VAR     Byte

' -----[ Main Routine ]-----------------------------------------------------

' Replace this DO...LOOP with your own Code.

DO
  GOSUB Get_Ir_Remote_Code
  DEBUG CLS, "Remote code: ", DEC remoteCode
  PAUSE 100
LOOP


' -----[ Subroutine - Get_Ir_Remote_Code ]---------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0                            ' Clear all bits in remoteCode.

  DO                                        ' Wait for rest between messages.
    RCTIME IrDet, 1, irPulse
  LOOP UNTIL irPulse > 1000

  PULSIN IrDet, 0, irPulse                  ' Measure pulse.
  IF irPulse > 500 THEN remoteCode.BIT0 = 1  ' Set (or leave clear) bit-0.
  RCTIME IrDet, 0, irPulse                  ' Measure next pulse.
  IF irPulse > 300 THEN remoteCode.BIT1 = 1  ' Set (or leave clear) bit-1.
  RCTIME IrDet, 0, irPulse                  ' etc.
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT4 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT5 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT6 = 1

  ' Adjust remoteCode so that keypad keys correspond to the value
  ' it stores.

  IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
  IF (remoteCode = 10) THEN remoteCode = 0

  RETURN
```

Come Funziona IrRemoteButtons.bs2

Questa dichiarazione **PIN** dà un nome ai piedini I/O che sentono l'uscita del rivelatore IR. Potete ora usare il nome **IrDet** al posto di **IN9**. Potete anche usarlo come argomento di un comando, così invece di **PULSIN 9, 0, IrPulse**, potete usare **PULSIN IrDet, 0, IrPulse**.

```
' -----[ I/O Definitions ]-------------------------------------------

' SONY TV IR remote declaration - input receives from IR detector

IrDet          PIN     9
```

Questi nomi di costanti possono essere usati al posto dei valori della tastiera. Questo vi permette di prendere decisioni sul valore dal telecomando infrarosso con nomi significativi come ad esempio **Enter** invece di 11 e **ChUp** invece di 16. Per esempio, se voleste prendere una decisione **IF…THEN** in base alla pressione del tasto ENTER, è molto meglio usare **IF (remoteCode = Enter) THEN** invece di **IF (remoteCode = 11) THEN**.

```
' -----[ Constants ]-------------------------------------------------

' SONY TV IR remote constants for non-keypad buttons

Enter          CON     11
ChUp           CON     16
ChDn           CON     17
VolUp          CON     18
VolDn          CON     19
Power          CON     21
```

Queste sono variabili che vi serviranno per le vostre subroutine. La variabile **time** da PulsesToByteValue.bs2 è stata rinominata **irPulse**.

```
' -----[ Variables ]-------------------------------------------------

' SONY TV IR remote variables

irPulse        VAR     Word
remoteCode     VAR     Byte
```

Molti esempi di applicazioni PBASIC pubblicate in cui potrete imbattervi hanno un semplice commento come per esempio **' Insert your code here**, eventualmente seguito da un comando **END**. In quel caso, sarà compito vostro leggere i commenti, le

subroutine, ed altre parti del programma per capire come farli funzionare. Spesso potrebbe esserci un altro programma disponibile per il download che spiega alcune delle cose che potete fare con l'applicazione. Questa main routine ha un commento per dove inserire il codice, ma ha anche un semplice ciclo **DO…LOOP** che vi permette di provarlo.

```
' -----[ Main Routine ]------------------------------------------------

' Replace this DO...LOOP with your own code.

DO
  GOSUB Get_Ir_Remote_Code
  DEBUG CLS, "Remote code: ", DEC remoteCode
  PAUSE 100
LOOP
```

Il blocco di codice nella subroutine **Get_Ir_Remote_Code** è per la maggior parte il contenuto del ciclo **DO…LOOP** del programma PulsesToByteValue.bs2 (dell' Attività #1). Ci sono due differenze. Primo, è in una subroutine invece che nel ciclo **DO…LOOP** della routine principale. Secondo, questa subroutine ha due dichiarazioni **IF…THEN** subito prima del comando **RETURN** che sono state studiate nell'ultima sezione Il Vostro Turno nell' Attività #1. Esse regolano il valore che la variabile **remoteCode** memorizza in modo che essa coincida con le cifre dei tasti che vengono premuti.

```
' -----[ Subroutine - Get_Ir_Remote_Code ]-------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0                              ' Clear all bits in remoteCode.

  DO                                          ' Wait for rest between messages.
    RCTIME IrDet, 1, irPulse
  LOOP UNTIL irPulse > 1000

  PULSIN IrDet, 0, irPulse                    ' Measure pulse.
  IF irPulse > 500 THEN remoteCode.BIT0 = 1   ' Set (or leave clear) bit-0.
  RCTIME IrDet, 0, irPulse                    ' Measure next pulse.
  IF irPulse > 300 THEN remoteCode.BIT1 = 1   ' Set (or leave clear) bit-1.
  RCTIME IrDet, 0, irPulse                    ' etc.
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT4 = 1
  RCTIME IrDet, 0, irPulse
```

```
IF irPulse > 300 THEN remoteCode.BIT5 = 1
RCTIME IrDet, 0, irPulse
IF irPulse > 300 THEN remoteCode.BIT6 = 1

' Adjust remoteCode so that keypad keys correspond to the value
' it stores.

IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
IF (remoteCode = 10) THEN remoteCode = 0

RETURN
```

Il Vostro Turno – Espandere l'Elenco dei Tasti Conosciuti

Potete espandere il vostro programma per includere i tasti SLEEP, MUTE, e LAST.

- √ Rilanciate TestIrRemoteButtons.bs2 ed acquisite i valori per i tasti SLEEP, MUTE e LAST.
- √ Salvate il programma come TestIrRemoteButtonsYourTurn.bs2.
- √ Modificate la sezione delle costanti in modo che questi valori siano compresi. State attenti, SLEEP è anche un comando PBASIC, quindi dovrete scegliere un nome diverso per il valore che corrisponde al tasto SLEEP sul telecomando. Provate invece **`FnSleep`**; è l'abbreviazione per la funzione-sleep.
- √ Modificate la dichiarazione **`SELECT…CASE`** in modo che visualizzi "SLEEP" quando viene premuto il tasto SLEEP, "MUTE" quando viene premuto il tasto MUTE, ed infine "LAST" quando viene premuto il tasto LAST. NOTA: in alcuni telecomandi il tasto LAST è chiamato PREV CH.

Tasti di controllo del VCR

I tasti di controllo del VCR come >> (FAST FORWARD) e << (REWIND) non fanno emettere codici dal telecomando quando è in modalità controllo TV. Potete usare il vostro telecomando e provare a programmarlo con i codici dei VCR SONY. Uno di questi di solito opera facendo funzionare il telecomando con la stessa codifica PWM dei controlli per i TV SONY. Ricordatevi che tutto ciò funziona con alcuni (non con tutti) i telecomandi universali.

I tasti per il VCR possono quindi essere usati per abilitare le funzioni di controllo del VCR. Premesso che abbiate programmato il codice giusto, la maggior parte dei tasti con le funzioni TV continueranno a funzionare. Potete usare IrRemoteButtons.bs2 per visualizzare i valori di **`remoteCode`** per ciascun tasto di controllo del VCR ed espandere il vostro elenco di costanti (la direttiva **`CON`**). I valori della variabile **`remoteCode`** per i tasti di controllo del VCR (STOP, PAUSE, PLAY, REWIND, FAST FORWARD, e RECORD) dovrebbero avere valori tra 24 e 29.

ATTIVITÀ #3: COLLAUDO DELL'APPLICAZIONE CON LA NAVIGAZIONE DEL BOE-BOT

Ci sono molti kit applicativi, note applicative ed articoli di mensili che mostrano come usare i moduli microcontrollori BASIC Stamp con tutti i tipi di sensori, attuatori e coprocessori. Potete usare molte di queste risorse per aggiungere funzioni al vostro Boe-Bot. Molti programmi esempio già scritti, in queste risorse, sono formattati in modo simile all'applicazione IrRemoteButtons.bs2 nell' Attività che avete appena terminato. Vedrete spesso dichiarazioni di costanti per significare numeri, subroutine che effettuano la funzione di riconoscere i tasti e vedrete anche le variabili necessarie per l'applicazione di queste subroutine. Potrete anche trovare altre sezioni che contengono direttive **`DATA`**, routine di inizializzazione, e storici di revisione.

In questa Attività, scriverete una routine principale di navigazione per il Boe-Bot che usa le funzioni supportate da IrRemoteButtons.bs2. familiarizzando con l'addattamento di questi programmi riusabili al vostro Boe-Bot, potrete usare una più vasta gamma di risorse pubblicate. Specialmente quando si ha a che fare con un nuovo sensore, display o processore audio, il lavoro di sgrossatura è già stato fatto e pubblicato nei programmi esempio. Starà a voi adattarlo alla vostra applicazione robotica (o altra applicazione). In molti casi, prenderete subroutine con le loro relative costanti e variabili da svariati programmi e li combineretee in una applicazione principale che controlla diversi sottosistemi.

Una Semplice Routine Principale per il Boe-Bot

Nell' Attività precedente, avete sostituito la routine principale in IrRemoteButtons.bs2 con un blocco di codice che conteneva una dichiarazione **`SELECT…CASE`**. Il modo migliore per ricordare come funziona **`SELECT…CASE`** è di usarlo per **`SELEZIONARE`** (una costante, variabile o espressione) e valutarla caso per caso. **`SELECT remoteCode`** rende possibile valutare la variabile **`remoteCode`** caso per caso, eseguendo diversi comandi **`DEBUG`** ciascuno per ciascun **`CASE`** (il valore memorizzato in **`remoteCode`**).

`SELECT…CASE` è anche molto adatto per la navigazione del Boe-Bot. Invece dei comandi **`DEBUG`**, ciascuna dichiarazione **`CASE`** può contenere blocchi di codice che inviano impulsi per manovre differenti del Boe-Bot. questo può essere messo in evidenza ripetendo le funzionalità di 4BitRemoteBoeBot.bs2 dal Capitolo 1, Attività #5. La differenza è, questa volta, che sarà estremamente facile scrivere ed anche di più modificare ed espandere.

Tutto quello che dovete fare con il prossimo programma esempio è aprire il programma IrRemoteButtons.bs2, salvarlo con un nuovo nome, e modificare la routine principale. Il codice per la lettura del telecomando IR viene inserito in una subroutine, quindi potrete dedicarvi alla programmazione del Boe-Bot, e tutto quello che dovrete sapere è il valore della variabile **`remoteCode`** dopo aver chiamato la subroutine **`Get_Ir_Remote_Code`**.

Programma Esempio – 7BitRemoteBoeBot.bs2

Questo esempio vi mette a disposizione tutti i tasti numerici di 4BitRemoteBoeBot.bs2 dal Capitolo 1, Attività #5, più i tasti CH+/- e VOL+/- del Capitolo 1, Attività #4.

√ Aprite IrRemoteButtons.bs2 e salvatelo con il nome 7BitRemoteBoeBot.bs2.

√ Aggiungete una sezione di inizializzazione subito prima della routine principale:

```
' -----[ Initialization ]--------------------------------------------

DEBUG "Press and hold a key (1-9 or CH/VOL)..."
FREQOUT 4, 2000, 3000                       ' Start/reset indicator.
```

√

√ Nella routine principale sostituite il ciclo **`DO…LOOP`** con questo ciclo:

```
DO

  ' Call subroutine that loads the IR message value into the
  ' remoteCode variable.

  GOSUB Get_Ir_Remote_Code

  ' Send PULSOUT durations for the various maneuvers based on
  ' the value of the remoteCode variable.

  SELECT remoteCode
    CASE 2, ChUp                            ' Forward
      PULSOUT 13, 850
      PULSOUT 12, 650
    CASE 4, VolDn                           ' Rotate Right
      PULSOUT 13, 650
      PULSOUT 12, 650
    CASE 6, VolUp                           ' Rotate Left
      PULSOUT 13, 850
      PULSOUT 12, 850
    CASE 8, ChDn                            ' Backward
      PULSOUT 13, 650
      PULSOUT 12, 850
    CASE 1                                  ' Pivot Fwd-left
      PULSOUT 13, 750
      PULSOUT 12, 650
```

```
      CASE 3                                    ' Pivot Fwd-right
        PULSOUT 13, 850
        PULSOUT 12, 750
      CASE 7                                    ' Pivot back-left
        PULSOUT 13, 750
        PULSOUT 12, 850
      CASE 9                                    ' Pivot back-right
        PULSOUT 13, 650
        PULSOUT 12, 750
      CASE ELSE                                 ' Hold position
        PULSOUT 13, 750
        PULSOUT 12, 750
    ENDSELECT

  LOOP
```

A seguire c'è una copia completa del programma.

- √ Lanciate e provate il progamma. Verificate che i tasti da 1 a 9 funzionano come previsto.
- √ Provate anche i tasti CH+/- e VOL +/-, e verificate che operino correttamente.
- √ Considerate come questo approccio è stato molto più semplice dell'approccio che è stato usato nel Capitolo 1, Attività #4 e #5.
- √ Salvate il vostro lavoro, salverete anche copie di questo programma per modificarle in seguito.

```
' -----[ Title ]-----------------------------------------------------------
' IR Remote for the Boe-Bot - 7BitRemoteBoeBot.bs2

' With an IR remote conFigurad to control a SONY TV, point the remote at
' the Boe-Bot and press and hold the 1-9 keys for different maneuvers.
' You can also use CH+/- and VOL+/-..

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ I/O Definitions ]--------------------------------------------------

' SONY TV IR remote declaration - input receives from IR detector

IrDet          PIN     9

' -----[ Constants ]-------------------------------------------------------

' SONY TV IR remote constants for non-keypad buttons

Enter          CON     11
ChUp           CON     16
```

```
ChDn            CON     17
VolUp           CON     18
VolDn           CON     19
Power           CON     21

' -----[ Variables ]-------------------------------------------------------

' SONY TV IR remote variables

irPulse         VAR     Word
remoteCode      VAR     Byte

' -----[ Initialization ]---------------------------------------------------

DEBUG "Press and hold a key (1-9 or CH/VOL)..."
FREQOUT 4, 2000, 3000                       ' Start/reset indicator.

' -----[ Main Routine ]-----------------------------------------------------

' Boe-Bot button control routine.

DO

  ' Call subroutine that loads the IR message value into the
  ' remoteCode variable.

  GOSUB Get_Ir_Remote_Code

  ' Send PULSOUT durations for the various maneuvers based on
  ' the value of the remoteCode variable.

  SELECT remoteCode
    CASE 2, ChUp                            ' Forward
      PULSOUT 13, 850
      PULSOUT 12, 650
    CASE 4, VolDn                           ' Rotate Left
      PULSOUT 13, 650
      PULSOUT 12, 650
    CASE 6, VolUp                           ' Rotate Right
      PULSOUT 13, 850
      PULSOUT 12, 850
    CASE 8, ChDn                            ' Backward
      PULSOUT 13, 650
      PULSOUT 12, 850
    CASE 1                                  ' Pivot Fwd-left
      PULSOUT 13, 750
      PULSOUT 12, 650
    CASE 3                                  ' Pivot Fwd-right
      PULSOUT 13, 850
      PULSOUT 12, 750
    CASE 7                                  ' Pivot Back-left
```

```
      PULSOUT 13, 750
      PULSOUT 12, 850
    CASE 9                                  ' Pivot Back-right
      PULSOUT 13, 650
      PULSOUT 12, 750
    CASE ELSE                               ' Hold Position
      PULSOUT 13, 750
      PULSOUT 12, 750
  ENDSELECT

LOOP

' -----[ Subroutine - Get_Ir_Remote_Code ]--------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0                            ' Clear all bits in remoteCode.

  DO                                        ' Wait for rest between messages.
    RCTIME IrDet, 1, irPulse
  LOOP UNTIL irPulse > 1000

  PULSIN IrDet, 0, irPulse                  ' Measure pulse.
  IF irPulse > 500 THEN remoteCode.BIT0 = 1  ' Set (or leave clear) bit-0.
  RCTIME IrDet, 0, irPulse                  ' Measure next pulse.
  IF irPulse > 300 THEN remoteCode.BIT1 = 1  ' Set (or leave clear) bit-1.
  RCTIME IrDet, 0, irPulse                  ' etc.
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT4 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT5 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT6 = 1

  ' Adjust remoteCode so that keypad keys correspond to the value
  ' it stores.

  IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
  IF (remoteCode = 10) THEN remoteCode = 0

  RETURN
```

Come Funziona 7BitRemoteBoeBot.bs2

La dichiarazione **SELECT…CASE** all'interno del ciclo **DO…LOOP** della routine principale è perfetto per inviare comandi **PULSOUT** in base al valore della variabile **remoteCode**.

```
GOSUB Get_Ir_Remote_Code

SELECT remoteCode
  CASE 2, ChUp                              ' Forward
    PULSOUT 13, 850
    PULSOUT 12, 650
  CASE 4, VolDn                             ' Rotate Left
    PULSOUT 13, 650
    PULSOUT 12, 650
      •
      •
      •
  CASE ELSE                                 ' Hold Position
    PULSOUT 13, 750
    PULSOUT 12, 750
ENDSELECT
```

Notare che le dichiarazioni **CASE** stanno usando costanti come **ChUp** e **VolDn**, dal modello della sezione di dichiarazione delle costanti di IrRemoteButtons.bs2:

```
' -----[ Constants ]-------------------------------------------------

' SONY TV IR remote constants for non-keypad buttons.

Enter           CON     11
ChUp            CON     16
ChDn            CON     17
VolUp           CON     18
VolDn           CON     19
Power           CON     21
```

Il Vostro Turno – Aggiungere il tasto POWER

Potete sfruttare il pulsante Power come interruttore di funzionalità del Boe-Bot. In altre parole, se il pulsante Power viene premuto, il Boe-Bot può essere programmato per cessare di rispondere ai comandi del telecomando. In questo esempio, il pulsante Reset sulla Board of Education deve essere premuto e rilasciato per far ripartire il programma.

- √ Nella vostra versione modificata di 7BitRemoteBoeBot.bs2 inserite una dichiarazione **CASE** per il pulsante POWER con un comando **END**.
- √ Provatelo e se necessario risolvete i problemi.

ATTIVITÀ #4: INSERIMENTO DI GRANDI NUMERI CON LA TASTIERA

Nella maggior parte delle apparecchiature potete inserire grandi numeri premendo sequenze di tasti numerici. Per esempio, potete premere 3-1-5 sulla tastiera del vostro microonde per riscaldare del cibo, ed il forno lo cuocerà per 3 minuti e 15 secondi. Similmente, potreste inserire 1-0-0-0 nella vostra calcolatrice come numero (mille) che deve essere moltiplicato, diviso, etc. molti apparecchi televisivi hanno una selezione tramite menù dove potete usare la tastiera del vostro telecomando per aggiornare l'ora, ed i VCR hanno la stessa caratteristica così che possiate programmare il VCR per registrare un programma TV in vostra assenza.

L'inserimento di cifre multiple è una caratteristica che può essere molto utile in un secondo esempio applicativo che potete aggiungere alla vostra libreria di programmi utili. In questa attività, svilupperete questa caratteristica, quindi la salverete per usi e riusi futuri.

Il Circuito dell' Altoparlante

Che sia una tastiera di un sistema antifurto, di un microonde, o di un Boe-Bot, un altoparlante aiuta realmente nel farvi sapere che il dispositivo ha capito che avete premuto quel tasto. L'altoparlante mostrato in Figura 2-6 è già stato aggiunto al vostro Boe-Bot nel Capitolo 1, Attività #4. ma fino ad ora, è stato usato solamente per avvertirvi che il programma è partito (o è ripartito spontaneamente a causa di pile scariche).

In questa attività, useremo anche il cicalino piezoelettrico per segnalare che il Boe-Bot ha rilevato e compreso quando avete premuto un tasto sul telecomando. Questo è specialmente utile quando digitate valori a cifre multiple sulla tastiera numerica del telecomando. Scriverete anche comandi che inviano dtoni differenti ad indicare che è stato premuto il tasto sbagliato.

- √ Se non lo avete già fatto, aggiungete il circuito altoparlante mostrato in Figura 2-6 nella vostra area prototipale del vostro Boe-Bot.

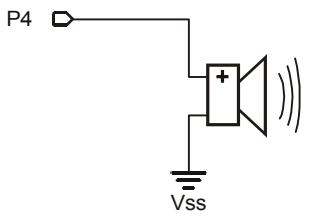


Figura 2-6
Altoparlante per l'avviso di Tasto Premuto

Conversione di Sequenze di Cifre in numeri Decimali

Diciamo che volete inviare al Boe-Bot un valore molto grande, come il valore decimale 635. per inviare questo numero, dovrete premere e rilasciare il 6, poi il 3, quindi il 5, per finire con il tasto ENTER. Il BASIC Stamp dovrà successivamente ricevere ciascuna cifra, moltiplicarla per 10, quindi aggiungere l'ultima cifra ricevuta. Quando viene premuto il tasto ENTER, viene inviato il valore 11, che significa che la la routine "moltiplica per dieci e somma" è finita. Questo è un elenco passo-passo di come il BASIC Stamp deve processare i messaggi in ingresso per ricostruire il valore 635:

- È stata ricevuta la cifra 6, quindi si memorizza 6.
- È stata ricevuta la cifra 3, quindi si moltiplica 6 per dieci, e si aggiunge il 3.
- È stata ricevuta la cifra 5, quindi si moltiplica 63 per dieci, e si aggiunge il 5.
- È stato ricevuto il valore 11, allora si memorizza il valore 635 e si esce dalla routine.

Progamma Esempio: EnterLargeValues.bs2

Questo programma esempio si comporta in modo simile ad un microonde con una tastiera. Potrete assemblare numeri più grandi, come 635 premendo e rilasciando i tasti 6, 3, e 5 seguiti dal tasto ENTER. Questo programma fà emettere dal cicalino piezodel Boe-Bot un tono per manifestare ogni pressione dei tasti. L'altoparlante fornisce anche la funzione antirimbalzo per i tasti.

Che cosa è "la Funzione Antirimbalzo"?

I circuiti elettronici operano ad una velocità molto maggiore delle azioni umane e dei contatti meccanici. I progettisti di circuiti e di sistemi integrati devono tener conto di questo fatto durante la progettazione di circuiti. Quando un interruttore viene chiuso od un pulsante viene premuto, c'è una collisione tra i contatti metallici. La superfice metallica tocca e rimbalza alcune volte prima di assestarsi e mantenere un contatto valido. Questo a sua volta viene visto dal processore come una rapida successione di uno e zero. Un circuito od una routine di programma che renda impossibile a questi zeri e uno di confondere il processore viene chiamato circuito **antirimbalzo**. Gli umani hanno anche la tendenza a premere un pulsante od un interruttore per un tempo individuale. Specialmente quando si inserisce la stessa cifra più volte, è importante programmare il microcontrollore per assicurarsi che vengano riconosciute le tendenze naturali durante la pressione dei pulsanti.

√ Aprite IrRemoteButtons.bs2, quindi salvatelo come EnterLargeValues.bs2.
√ Aggiungete questa direttiva alla sezione Definizioni I/O:

```
Speaker        PIN     4
```

√ Aggiungete questa variabile alla sezione Variabili del programma.

```
' Main Routine Variables

value          VAR     Word                  ' Stores multi-digit value
```

√ Nella routine principale sostituite il ciclo **DO…LOOP** con questo:

```
DO

  DEBUG "Type a value (up to 65535)", CR, "Then press ENTER", CR, CR

  value = 0
  remoteCode = 0
  DEBUG "Digits entered: "

  DO

    value = value * 10 + remoteCode

    GOSUB Get_Ir_Remote_Code

    IF (remoteCode >= 0) AND (remoteCode <= 9) THEN
      DEBUG DEC remoteCode
    ENDIF

    FREQOUT Speaker, 100, 3500
    PAUSE 200
```

```
    LOOP UNTIL (remoteCode = Enter)

    DEBUG CR, "The value is: ", DEC value, CR, CR

LOOP
```

- √ Seguite le richieste del Terminale di Debug. E digitate un numero sulla tastiera del telecomando come se di stesse digitando dei numeri sulla tastiera di una calcolatrice.
- √ Premere e rilasciare il tasto ENTER.
- √ Verificate che il valore memorizzato dal BASIC Stamp sia corretto.
- √ Per inviare, e memorizzare, numeri con diverse cifre al BASIC Stamp usate i tasti numerici ed il tasto ENTER della tastiera.

Come Funziona EnterLargeValues.bs2

Questo programma origina da IrRemoteButtons.bs2. e sono state modificate soltanto le sezioni Definizioni I/O, Variabili, e la Routine Principale.

- √ Viene dichiarata una variabile word di nome **`value`** in aggiunta alle variabili per la decodifica del messaggio IR remoto. Questa variabile memorizza il valore a più cifre acquisito dal telecomando. La direttiva **`Speaker PIN 4`**, vi permette di usare **`Speaker`** come argomento ***`Pin`*** del comando **`FREQOUT`**.

```
' SONY TV IR remote variables

irPulse        VAR     Word
remoteCode     VAR     Byte

' Main Routine variables

value          VAR     Word                  ' Stores multi-digit value
```

la prima cosa effettuata dalla Routine Principale è l'azzeramento delle variabili **`value`** e **`remoteCode`**. Quindi, un comando **`DEBUG`** visualizza il messaggio "Digits entered: ".

```
    value = 0
    remoteCode = 0
    DEBUG "Digits entered: "
```

Il ciclo **`DO…LOOP`** per l'ingresso dalla tastiera è condizionale, e viene eseguito fino a che non viene premuto il tasto ENTER. Man mano che le cifre vengono inviate in successione, la variabile **`value`** viene moltiplicata per dieci, quindi viene sommata al valore la cifra

più recente memorizzata da **remoteCode**. Sostanzialmente è ciò che fà **value = value * 10 + remoteCode**. Notate che un comando **FREQOUT** ed un comando **PAUSE** seguono il comando **GOSUB Get_Ir_Remote_Code**. Questo fà emettere un bip al Boe-Bot ad ogni pressione di tasto. Il bip e la pausa sono sufficientemente lunghi per far sì che l'utente lrilasci il tasto. Questo evita che una singola pressione del tasto 5 sia ricevuto come valore 555.

```
DO

  value = value * 10 + remoteCode

  GOSUB Get_Ir_Remote_Code

  IF (remoteCode >= 0) AND (remoteCode <= 9) THEN
    DEBUG DEC remoteCode
  ENDIF

  FREQOUT Speaker, 100, 3500
  PAUSE 200

LOOP UNTIL (remoteCode = Enter)
```

Dopo che il tasto ENTER è stato premuto e rilasciato, il ciclo **DO…LOOP** termina, e viene visualizzato il numero a più cifre memorizzato nella variabile **value**:

```
DEBUG "The value is: ", DEC value, CR, CR
```

Il Vostro Turno – Processare Solamente le Cifre ed il tasto Enter

Se state progettando un prodotto con il BASIC Stamp ed un telecomando universale, se usate EnterLargeValues.bs2 così com'è, potreste ricevere delle lamentele dai clienti. Il programma deve ignorare i tasti non numerici come POWER e VOL+. Premere questi tasti può portare ad alcuni valori abbastanza strani.

- √ Provate a digitare cifre non numeriche con la vostra versione originale e non modificata di EnterLargeValues.bs2.
- √ Salvate il programma con il nome di EnterLargeValuesYourTurn.bs2.
- √ Sostituite il ciclo **DO…LOOP** nella routine principale con questo:

```
DO

  DEBUG "Type a value (up to 65535)", CR, "Then press ENTER", CR, CR

  value = 0
```

```
  remoteCode = 0

  DO

    value = value * 10 + remoteCode

    DO
      GOSUB Get_Ir_Remote_Code
      IF (remoteCode < 10) THEN
        DEBUG "You pressed: ", DEC1 remoteCode, CR
        GOSUB Beep_Valid
        EXIT
      ELSEIF (remoteCode = Enter) THEN
        DEBUG "You pressed: ENTER", CR
        GOSUB Beep_Valid
        EXIT
      ELSE
        DEBUG "Press 0-9 or ENTER", CR
        GOSUB Beep_Error
      ENDIF
    LOOP

  LOOP UNTIL (remoteCode = Enter)

  DEBUG CR, "The value is: ", DEC value, CR, CR

LOOP
```

√ Aggiungete queste due subroutine alla fine del vostro programma:

```
' -----[ Subroutine - Beep_Valid ]-----------------------------------

' Call this subroutine to acknowledge a key press.

Beep_Valid:

  FREQOUT Speaker, 100, 3500
  PAUSE 200

  RETURN

' -----[ Subroutine - Beep_Error ]-----------------------------------

' Call this subroutine to reject a key press.

Beep_Error:

  FREQOUT Speaker, 100, 3000
  PAUSE 200

  RETURN
```

- √ Provate questa soluzione e verificate che funzioni.
- √ Salvate il vostro lavoro.

Aggiungere la funzione di Inserimento da Tastiera per la vostra Applicazione

Copiando la parte funzionale di EnterLargeValues.bs2 in una subroutine di IrRemoteButtons.bs2, avrete un nuovo programma applicativo che potrà sia leggere la presssione di un singolo tasto da un telecomando remoto che l'acquisizione di valori grandi tramite la tastiera. Quale funzione verrà eseguita dipende da quale subroutine viene chiamata.

Programma Esempio – IrRemoteKeypad.bs2

- √ Aprire EnterLargeValuesYourTurn.bs2 e salvare in IrRemoteKeypad.bs2.
- √ Aggiungere una subroutine chiamata **`Get_Multi_Digit_Value`** in IrRemoteKeypad.bs2.
- √ Usate Edit → Copy ed Edit → Paste per muovere ogni cosa mostrata sotto nel programma EnterLargeValuesYourTurn.bs2 nella nuova subroutine **`Get_Multi_Digit_Value`** in IrRemoteKeypad.bs2.

```
value = 0
remoteCode = 0

DO

  value = value * 10 + remoteCode

  DO
    GOSUB Get_Ir_Remote_Code
    IF (remoteCode < 10) THEN
      DEBUG "You pressed: ", DEC1 remoteCode, CR
      GOSUB Beep_Valid
      EXIT
    ELSEIF (remoteCode = Enter) THEN
      DEBUG "You pressed: ENTER", CR
      GOSUB Beep_Valid
      EXIT
    ELSE
      DEBUG "Press 0-9 or ENTER", CR
      GOSUB Beep_Error
    ENDIF
  LOOP

LOOP UNTIL (remoteCode = Enter)
```

- √ Aggiungete un comando **RETURN** alla fine della subroutine.
- √ Modificate la routine principale in modo simile al seguente:

```
' Replace this DO...LOOP with your own code.

DO
  DEBUG "Enter a value: "
  GOSUB Get_Multi_Digit_Value
  DEBUG "The value is: ", DEC value, CR, CR
LOOP
```

- √ Aggiungete commenti al vostro programma spiegando come funziona la dichiarazione **DO…LOOP UNTIL**.

Dopo che avete fatto questo cambiamenti, il vostro programma dovrebbe assomigliare a quello sotto riportato.

- √ Lanciate e provate IrRemoteKeypad.bs2.
- √ Risolvete i problemi se ce ne sono.
- √ Salvate il programma dopo che lo avete interamente controllato e provato.

```
' -----[ Title ]-----------------------------------------------------------
' IR Remote for the Boe-Bot - IrRemoteKeypad.bs2
' Capture and store button codes sent by a universal remote conFigurad to
' control a SONY TV.  This program also supports keypad entry of
' multi-digit values.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ I/O Definitions ]--------------------------------------------------

' SONY TV IR remote declaration - input receives from IR detector

IrDet          PIN     9
Speaker        PIN     4

' -----[ Constants ]-------------------------------------------------------

' SONY TV IR remote constants for non-keypad buttons.

Enter          CON     11
ChUp           CON     16
ChDn           CON     17
VolUp          CON     18
VolDn          CON     19
Power          CON     21
```

```
' -----[ Variables ]------------------------------------------------------

' SONY TV IR remote variables

irPulse        VAR     Word                  ' Single-digit remote variables
remoteCode     VAR     Byte
index          VAR     Nib
value          VAR     Word                  ' Stores multi-digit value

' -----[ Main Routine ]----------------------------------------------------

' Replace this DO...LOOP with your own code.

DO
  DEBUG "Enter a value: ", CR
  GOSUB Get_Multi_Digit_Value
  DEBUG "The value is: ", DEC value, CR, CR
LOOP

' -----[ Subroutine - Get_Ir_Remote_Code ]--------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0                            ' Clear all bits in remoteCode.

  DO                                        ' Wait for rest between messages.
    RCTIME IrDet, 1, irPulse
  LOOP UNTIL irPulse > 1000

  PULSIN IrDet, 0, irPulse                  ' Measure pulse.
  IF irPulse > 500 THEN remoteCode.BIT0 = 1  ' Set (or leave clear) bit-0.
  RCTIME IrDet, 0, irPulse                  ' Measure next pulse.
  IF irPulse > 300 THEN remoteCode.BIT1 = 1  ' Set (or leave clear) bit-1.
  RCTIME IrDet, 0, irPulse                  ' etc.
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT4 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT5 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT6 = 1

  ' Adjust remoteCode so that keypad keys correspond to the value
  ' it stores.
```

```
  IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
  IF (remoteCode = 10) THEN remoteCode = 0

  RETURN

' -----[ Subroutine - Get_Multi_Digit_Value ]-----------------------------

' Acquire multi-digit value (up to 65535) and store it in
' the value variable.  Speaker beeps each time a key is
' pressed.

Get_Multi_Digit_Value:

  value = 0
  remoteCode = 0

  DO

    value = value * 10 + remoteCode

    DO
      GOSUB Get_Ir_Remote_Code
      IF (remoteCode < 10) THEN
        DEBUG "You pressed: ", DEC1 remoteCode, CR
        GOSUB Beep_Valid
        EXIT
      ELSEIF (remoteCode = Enter) THEN
        DEBUG "You pressed: ENTER", CR
        GOSUB Beep_Valid
        EXIT
      ELSE
        DEBUG "Press 0-9 or ENTER", CR
        GOSUB Beep_Error
      ENDIF
    LOOP

  LOOP UNTIL (remoteCode = Enter)

  RETURN

' -----[ Subroutine - Beep_Valid ]-----------------------------------

' Call this subroutine to acknowledge a key press.

Beep_Valid:

  FREQOUT Speaker, 100, 3500
  PAUSE 200

  RETURN
```

```
' -----[ Subroutine - Beep_Error ]----------------------------------
' Call this subroutine to reject a key press.

Beep_Error:

  FREQOUT Speaker, 100, 3000
  PAUSE 200

  RETURN
```

Il Vostro Turno – Fare Copie di Backup delle Vostre Applicazioni

Userete e riuserete i programmi applicativi che avete sviluppato dalla Attività #2 alla Attività #4.

- √ Create una cartella separata per i vostri programmi applicativi.
- √ Fate copie di backup di IrRemoteButtons.bs2, 7BitRemoteBoeBot.bs2, e IrRemoteKeypad.bs2.

ATTIVITÀ #5: DIREZIONE E DISTANZA DEL BOE-BOT A TASTIERA

In questa attività, programmerete il Boe-Bot per ricevere le informazioni di direzione e distanza da un telecomando remoto ad infrarossi, ecco come funzionerà il programma:

- Premere/rilasciare un tasto CH o VOL per selezionare una di quattro manovre: avanti, indietro, ruota a destra, ruota a sinistra.
- Poi, usate la tastiera per inserire il numero di impulsi da inviare.
- Quando viene premuto/rilasciato il tasto ENTER, il Boe-Bot esegue le manovre.

Scrivendo una routine principale per il Boe-Bot main routine per IrRemoteKeypad.bs2, potrete fare uso dei codici di questi tasti o della funzione di inerimento con la tastiera. Questo porterà alla scrittura di una routine principale che vi permette di scegliere la direzione con i tasti CH e VOL quindi inserire un numero di impulsi con la tastiera molto più semplice nello sviluppo. Dovrete chiamare la subroutine **`Get_Ir_Remote_Code`** per attivare i tasti CH/VOL per la direzione. Dopo di chè, potrete chiamare la subroutine **`Get_Multi_Digit_Value`** per ottenere il numero degli impulsi.

Programma Esempio – KeypadDirectionDistance.bs2

- √ Aprite IrRemoteKeypad.bs2 e salvatelo come KeypadDirectionDistance.bs2.
- √ Aggiungete queste direttive **`PIN`**:

```
' Boe-Bot Servo Pins

ServoLeft      PIN     13
ServoRight     PIN     12
```

√ Aggiungete questa dichiarazione nella sezione delle Variabili.

```
' Boe-Bot navigation variables

direction      VAR     Byte
counter        VAR     Byte
```

√ Aggiungete la routine di inizializzazione per l'indicazione di start/reset del Boe-Bot insieme con alcune istruzioni operative per il Terminale di Debug. Questa sezione dovrebbe essere inserita subito prima della sezione Routine Principale.

```
' -----[ Initialization ]----------------------------------------------

DEBUG "Program Starting...", CR, CR           ' Start/reset indicator.

FREQOUT Speaker, 2000, 3000

DEBUG "Use CH/VOL for direction,", CR,        ' Debug instructions.
      "then type distance (up to ", CR,
      "255) then press ENTER.", CR, CR
```

√ Sostituite il codice della sezione Routine Principale con:

```
DO

  DEBUG "Select direction (CH/VOL):", CR

  DO

    GOSUB Get_Ir_Remote_Code

    IF (remoteCode < ChUp) OR (remoteCode > VolDn) THEN
      DEBUG "Select direction (CH/VOL):", CR
      GOSUB Beep_Error
    ENDIF

  LOOP UNTIL (remoteCode >= ChUp) AND (remoteCode <= VolDn)

  direction = remoteCode

  GOSUB Beep_Valid

  DEBUG "Enter number of pulses: ", CR

  GOSUB Get_Multi_Digit_Value
```

```
    DEBUG "The value is: ", DEC value, CR
    DEBUG "Running...", CR, CR

    FOR counter = 1 TO value

      SELECT direction
        CASE ChUp
          PULSOUT ServoLeft, 850
          PULSOUT ServoRight, 650
        CASE ChDn
          PULSOUT ServoLeft, 650
          PULSOUT ServoRight, 850
        CASE VolUp
          PULSOUT ServoLeft, 850
          PULSOUT ServoRight, 850
        CASE VolDn
          PULSOUT ServoLeft, 650
          PULSOUT ServoRight, 650
      ENDSELECT

      PAUSE 20

    NEXT

  LOOP
```

Un listto completo del programma è incluso al termine delle istruzioni elenco di verifica.

- √ Salvate quindi lanciate il vostro programma modificato.
- √ Assicuratevi l'interruttore a tre posizioni del vostro Boe-Bot sia in posizione 2.
- √ Premere/rilasciare il tasto CH+ per selezionare avanti.
- √ Premere/rilasciare i numeri 1, 6, 2.
- √ Premere/rilasciare il tasto ENTER.

Il Boe-Bot dovrebbe andare avanti per circa un metro (o yarda se state pensando in sistema anglosassone).

- √ Premere/rilasciare il tasto VOL+ per selezionare ruota a destra.
- √ Premere/rilasciare i numeri 2, 0.
- √ Premere/rilasciare il tasto ENTER.

Il Boe-Bot dovrebbe ruotare per circa 90-gradi in senso orario (verso destra).

- √ Per un ripasso circa impulsi e distanze, vedere *Robotica con il Boe-Bot* al Capitolo 4.
- √ Far pratica dell'invio di direzioni e distanze fino a che non abbiate sufficiente dimestichezza con la navigazione del vostro Boe-Bot.

```
' -----[ Title ]-----------------------------------------------------------
' IR Remote for the Boe-Bot - KeypadDirectionDistance.bs2
' Each Boe-Bot maneuver involves three Passos:
' 1) Select a maneuver
'    CH+ = Forward, CH- = Backward, VOL+ = Right, VOL- = Left
' 2) Type in a distance (1 to 255) pulses.
' 3) Press ENTER.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ I/O Definitions ]--------------------------------------------------

' SONY TV IR remote declaration - input receives from IR detector

IrDet          PIN     9
Speaker        PIN     4

' Boe-Bot Servo Pins

ServoLeft      PIN     13
ServoRight     PIN     12

' -----[ Constants ]-------------------------------------------------------

' SONY TV IR remote constants for non-keypad buttons.

Enter          CON     11
ChUp           CON     16
ChDn           CON     17
VolUp          CON     18
VolDn          CON     19
Power          CON     21

' -----[ Variables ]-------------------------------------------------------

' SONY TV IR remote variables

irPulse        VAR     Word                ' Single-digit remote variables
remoteCode     VAR     Byte
value          VAR     Word                ' Stores multi-digit value

' Boe-Bot navigation variables
```

```
direction      VAR     Byte
counter        VAR     Byte

' -----[ Initialization ]--------------------------------------------------

DEBUG "Program Starting...", CR, CR           ' Start/reset indicator.

FREQOUT Speaker, 2000, 3000

DEBUG "Use CH/VOL for direction,", CR,        ' Debug instructions.
      "then type distance (up to ", CR,
      "255) then press ENTER.", CR, CR

' -----[ Main Routine ]-----------------------------------------------------

DO

  DEBUG "Select direction (CH/VOL):", CR

  DO

    GOSUB Get_Ir_Remote_Code

    IF (remoteCode < ChUp) OR (remoteCode > VolDn) THEN
      DEBUG "Select direction (CH/VOL):", CR
      GOSUB Beep_Error
    ENDIF

  LOOP UNTIL (remoteCode >= ChUp) AND (remoteCode <= VolDn)

  direction = remoteCode

  GOSUB Beep_Valid

  DEBUG "Enter number of pulses: ", CR

  GOSUB Get_Multi_Digit_Value

  DEBUG "The value is: ", DEC value, CR
  DEBUG "Running...", CR, CR

  FOR counter = 1 TO value

    SELECT direction
      CASE ChUp
        PULSOUT ServoLeft, 850
        PULSOUT ServoRight, 650
      CASE ChDn
        PULSOUT ServoLeft, 650
        PULSOUT ServoRight, 850
      CASE VolUp
```

```
        PULSOUT ServoLeft, 850
        PULSOUT ServoRight, 850
      CASE VolDn
        PULSOUT ServoLeft, 650
        PULSOUT ServoRight, 650
    ENDSELECT

    PAUSE 20

  NEXT

LOOP

' -----[ Subroutine - Get_Ir_Remote_Code ]---------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0                              ' Clear all bits in remoteCode.

  DO                                          ' Wait for rest between messages.
    RCTIME IrDet, 1, irPulse
  LOOP UNTIL irPulse > 1000

  PULSIN IrDet, 0, irPulse                    ' Measure pulse.
  IF irPulse > 500 THEN remoteCode.BIT0 = 1   ' Set (or leave clear) bit-0.
  RCTIME IrDet, 0, irPulse                    ' Measure next pulse.
  IF irPulse > 300 THEN remoteCode.BIT1 = 1   ' Set (or leave clear) bit-1.
  RCTIME IrDet, 0, irPulse                    ' etc.
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT4 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT5 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT6 = 1

  ' Adjust remoteCode so that keypad keys correspond to the value
  ' it stores.

  IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
  IF (remoteCode = 10) THEN remoteCode = 0

  RETURN

' -----[ Subroutine - Get_Multi_Digit_Value ]-----------------------------
```

```
' Acquire multi-digit value (up to 65535) and store it in
' the value variable.  Speaker beeps each time a key is
' pressed.

Get_Multi_Digit_Value:

  value = 0
  remoteCode = 0

  DO

    value = value * 10 + remoteCode

    DO
      GOSUB Get_Ir_Remote_Code
      IF (remoteCode < 10) THEN
        DEBUG "You pressed: ", DEC1 remoteCode, CR
        GOSUB Beep_Valid
        EXIT
      ELSEIF (remoteCode = Enter) THEN
        DEBUG "You pressed: ENTER", CR
        GOSUB Beep_Valid
        EXIT
      ELSE
        DEBUG "Press 0-9 or ENTER", CR
        GOSUB Beep_Error
      ENDIF
    LOOP

  LOOP UNTIL (remoteCode = Enter)

  RETURN

' -----[ Subroutine - Beep_Valid ]-------------------------------------

' Call this subroutine to acknowledge a key press.

Beep_Valid:

  FREQOUT Speaker, 100, 3500
  PAUSE 200

  RETURN

' -----[ Subroutine - Beep_Error ]-------------------------------------

' Call this subroutine to reject a key press.

Beep_Error:

  FREQOUT Speaker, 100, 3000
```

```
PAUSE 200

RETURN
```

Come Funziona KeypadDirectionDistance.bs2

A IrRemoteKeypad.bs2 devono essere aggiunte due variabili, una per la memorizzazione della direzione del Boe-Bot, e l'altra per contare il numero di impulsi di un ciclo **FOR…NEXT**.

```
' Boe-Bot navigation variables

direction      VAR     Byte
counter        VAR     Byte
```

Il primo comando interno al ciclo **DO…LOOP** è un comando **DEBUG** che vi richiede una direzione con CH/VOL.

```
DEBUG "Select direction (CH/VOL):", CR
```

Il programma richiama continuamente la subroutine **Get_Ir_Remote_Code,** fino a che non viene riccevuto dal telecomando un **remoteCode** che rientra tra **ChUp** (16) e **VolDn** (19). Se qualche tasto che non sia CH+/- o VOL+/-, la dichiarazione **IF…THEN** invia un tono basso di errore insieme con una richiesta dal Terminale di Debug di premere uno dei tasti CH o VOL.

```
DO

  GOSUB Get_Ir_Remote_Code

  IF (remoteCode < ChUp) OR (remoteCode > VolDn) THEN
    DEBUG "Select direction (CH/VOL):", CR
    GOSUB Beep_Error
  ENDIF

LOOP UNTIL (remoteCode >= ChUp) AND (remoteCode <= VolDn)
```

Dopo che è stata chiamata la subroutine **Get_Ir_Remote_Code**, la direzione che vlte inviare al Boe-Bot con il telecomando è memorizzatanella variabile **remoteCode**. Questo valore deve essere memorizzato in una variabile diversa prima che un altro messaggio IR venga processato; altrimenti, il valore verrà perso. Ecco perchè il valore di direzione memorizzato in **remoteCode** deve essere copiato in un altra variabile, che è a questo scopo chiamata **direction**.

```
direction = remoteCode
```

Il tasto di destra deve essere premuto per far uscire il programma dal ciclo **DO…LOOP** che richiama la routine **Get_Ir_Remote_Code**. Il programma chiama la subroutine **Beep_Valid**, che fà emettere il tono acuto di riconoscimento della pressione del tasto corretto.

```
GOSUB Beep_Valid
```

Un altro comando **DEBUG** vi richiede di inserire il numero degli impulsi (usando la tastiera numerica del telecomando).

```
DEBUG "Enter number of pulses: ", CR
```

La chiamata alla subroutine **Get_Multi_Digit_Value** è la parte dove viene usata la tastiera numerica per digitare il numero di impulsi da inviare. Questa è anche la parte dove il valore di **remoteCode** cambia, che è il motivo per cui la direzione che avevate inserito deve essere copiata in un altra variabile. A differenza della variabile **remoteCode**, la variabile **value** non verrà sovrascritta prima che gli impulsi siano inviati ai servo. Quindi **value** non deve essere copiata in un altra variabile; può memorizzare il numero degli impulsi.

```
GOSUB Get_Multi_Digit_Value
```

Un comando **DEBUG** vi permette anche di verificare il valore inserito (se lasciate il Boe-Bot collegato al cavo di programmazione). Un secondo comando **DEBUG,** mentre ai servo vengono inviati gli imuplsi, visualizza il messaggio "Running…".

```
DEBUG "The value is: ", DEC value, CR
DEBUG "Running...", CR, CR
```

La direzione è ora memorizzata nella variabile **direction**, ed il numero degli impulsi è memorizzato nella variabile **value**. Un ciclo **FOR…NEXT** usa la variabile **value** per determinare quanti impulsi devono essere inviati ai servo. All'interno del ciclo **FOR…NEXT**, una dichiarazione **SELECT…CASE** usa il valore memorizzato nella variabile **direction** per decidere di inviare ai servo impulsi di quale durata. Dopo il comando **SELECT…CASE**, **PAUSE 20** mantiene costante l'intervallo tra gli impulsi dei servo.

```
FOR counter = 1 TO value

  SELECT direction
    CASE ChUp
      PULSOUT ServoLeft, 850
```

```
      PULSOUT ServoRight, 650
    CASE ChDn
      PULSOUT ServoLeft, 650
      PULSOUT ServoRight, 850
    CASE VolUp
      PULSOUT ServoLeft, 850
      PULSOUT ServoRight, 850
    CASE VolDn
      PULSOUT ServoLeft, 650
      PULSOUT ServoRight, 650
  ENDSELECT

  PAUSE 20

 NEXT
```

Il Vostro Turno – Ripetere l'Azione "LAST"

Per il controllo della TV, il tasto LAST (in qualche caso etichettato PREV CH) vi riporta indietro al canale che stavate vedendo prima del canale che state attualmente vedendo. Il tasto LAST può essere vantaggiosamente impiegato. Questo è una maniera di modificare il vostro programma per eseguire questo compito.

√ Salvate KeypadDirectionDistance.bs2 con il nuovo nome di KeypadDirectionDistanceYourTurn.bs2.

√ La sezione Il Vostro Turno dell' Attività #2 è consistita nell'espansione dll'elenco di direttive **CON** per i tasti del telecomando IR. Ecco una costante che dovrete aggiungere al vostro programma per il tasto LAST:

```
Last           CON     59
```

√ Modificate questo ciclo **DO…LOOP**:

```
DO

  GOSUB Get_Ir_Remote_Code

  IF (remoteCode < ChUp) OR (remoteCode > VolDn) THEN
    DEBUG "Select direction (CH/VOL):", CR
    GOSUB Beep_Error
  ENDIF

LOOP UNTIL (remoteCode >= ChUp) AND (remoteCode <= VolDn)
```

Aggiungendo una condizione alla dichiarazione **IF…THEN** che causi al programma il salto ad una etichetta chiamata **Servo_Pulses** se **remoteCode** memorizza il valore della costante **Last**.

```
DO

  GOSUB Get_Ir_Remote_Code

  IF (remoteCode) = Last THEN
    GOSUB Beep_Valid
    GOTO Servo_Pulses
  ELSEIF (remoteCode < ChUp) OR (remoteCode > VolDn) THEN
    DEBUG "Select direction (CH/VOL):", CR
    GOSUB Beep_Error
  ENDIF

LOOP UNTIL (remoteCode >= ChUp) AND (remoteCode <= VolDn)
```

√ Aggiungete questa etichetta **Servo_Pulses:** tra i due comandi **DEBUG** mostrati di seguito:

```
  DEBUG "The value is: ", DEC value, CR

Servo_Pulses:                              ' <--- Add this label.

  DEBUG "Running...", CR, CR
```

Attenzione al Codice Spaghetti (Confuso)!

Molti istruttori, programmatori di computer, direttori di progetto di robot ed altri spesso aggrottano le sopracciglia quando si usa un comando **GOTO** per saltare ad una etichetta da qualche altra parte del programma. La ragione per cui vengono chiamati "codice spaghetti" è data dalla difficoltà che si incontra cercando di trovare un errore in un programma con troppi comandi **GOTO**. Capire come funziona il programma, diventa come seguire visualmente un singolo spaghetto dentro un piatto.

Un modo di implementare il tasto LAST con codice non-spaghetti è di spostare il blocco di codice che invia impulsi ai servo in una subroutine. In questo modo si può usare un comando a **GOSUB** al posto di un comando **GOTO**. I comandi **GOSUB** tendono a semplificare il codice-spaghetti perchè il comando **RETURN** reinvia il programma al comando che segue immediatamente la chiamata alla subroutine.

√ Salvate KeypadDirectionDistanceYourTurn.bs2 con il nuovo nome di KeypadDirectionDistanceYourTurn2.bs2.

√ Muovete l'etichetta **`Servo_Pulses:`** label, il comando **`DEBUG`**, ed il ciclo **`FOR…NEXT`** che invia gli impulsi ai servo dalla routine principale ad una subroutine. Quando lo avrete fatto dovrebbe somigliare al seguente.

```
' -----[ Subroutine - Servo_Pulses ]---------------------------------

' Call this subroutine to deliver pulses to the servos.
' You must store the number of pulses in the value variable
' and the maneuver in the direction variable.  ChUp = forward,
' ChDn = backward, VolUp = rotate right, VolDn = rotate left.

Servo_Pulses:

  DEBUG "Running...", CR, CR

  FOR counter = 1 TO value

    SELECT direction
      CASE ChUp
        PULSOUT ServoLeft, 850
        PULSOUT ServoRight, 650
      CASE ChDn
        PULSOUT ServoLeft, 650
        PULSOUT ServoRight, 850
      CASE VolUp
        PULSOUT ServoLeft, 850
        PULSOUT ServoRight, 850
      CASE VolDn
        PULSOUT ServoLeft, 650
        PULSOUT ServoRight, 650
    ENDSELECT

    PAUSE 20

  NEXT

  RETURN
```

√ Al posto di quel codice appena rimosso dalla routine principale, aggiungete questa chiamata alla subroutine:

```
GOSUB Servo_Pulses
```

√ Cambiate questa riga nella routine principale:

```
  GOTO Servo_Pulses
```

In modo che sia:

```
GOSUB Servo_Pulses
```

√ Salvate, lanciate, e provate il programma. Verificate che si comporti nello stesso modo dell'implementazione con il comando **GOTO**.

SOMMARIO

La decodifica è il processo di conversione di un segnale elettronico in qualche cosa di comprensibile ed usabile. Nel caso del messaggio remoto dal telecomando IR, la decodifica consiste nella conversione della misura della durata degli impulsi in valori binari 1 e 0 in una variabile byte. Ogni volta che il processo si completa, la variabile byte memorizza un numero (codice) che corrisponde con un tasto della tastiera.

Per comprendere il processo di decodifica, sono stati studiati i concetti del conteggio binario e della conversione da binario a decimale. È stato studiato il modificatore **`.BIT`** come modo per impostare ed azzerare singoli bit in una variabile. Sono state scritte dichiarazioni **`IF…THEN`** che esaminano misure di impulsi per determinare il valore di un bit in una variabile che memorizza il valore decodificato della larghezza dell'impulso modulato in larghezza del messaggio del telecomando. Queste dichiarazioni **`IF…THEN`** impiegano il modificatore **`.BIT`** per impostare un bit di una variabile se la misura dell'impulso corrispondente era maggiore di un certo valore, o azzerare il bit se la misura dell'impulso era minore di un certo valore.

Questo capitolo ha sviluppato programmi applicativi con costanti, dichiarazione di variabili, e subroutine che servono come mattoni che potrete usare in programmi più grandi. Questi programmi applicativi hanno ridotto i programmi del capitolo 1 in qualche modo complicati a poche righe nella routine principale dell'applicazione.

Sono state studiate tecniche per la modifica di programmi applicativi esempio e per l'uso delle loro funzioni a vantaggio del Boe-Bot. Gli esempi inclusi usano le costanti con nomi utili nel prendere decisioni, il richiamo delle subroutine per acquisire i messaggi IR, l'aggiunta di dichiarazioni delle variabili, le direttive **`PIN`** e le routine che adattano il programma applicativo per l'uso con i servo del Boe-Bot.

Domande

1. Che cosa significa "decodificare" un messaggio IR di un telecomando universale?
2. Come viene normalmente chiamata una cifra binaria?
3. Come potete calcolare il valore che la particolare posizione di un bit rappresenta?
4. che cosa significa impostare o azzerare un bit?

5. Come si usa **`.BIT`** per impostare o azzerare i bit di una variabile?
6. Come vengono usate le costanti in IrRemoteButtons.bs2?
7. Che cosa significa quando si applica la funzione antirimbalzo ad un tasto?
8. Perche usereste un altoparlante con un pulsante o con una tastiera?

Esercizi

1. Contare da 0 a 7 in binario.
2. Come deve essere 24 in binario?
3. Calcolare il moltiplicatore che usereste per il 12° bit in un numero binario.
4. Convertire 1111 in un numero decimale.
5. Assumete di avere otto misure di impulsi da decodificare invece di sette. Spiegate come modificare la subroutine **`Get_Ir_Remote_Code`** per acquisire e decodificare la misura dell'impulso.
6. Diciamo che avete dichiarato una variabile bit chiamata **`onOffState`**. Scrivete un comando **`SELECT…CASE`** che cambi **`onOffState`** ad 1 se era 0 ed a 0 se era 1. Usate l'operatore **~** (not).

Progetto

1. Modificate 7BitRemoteBoeBot.bs2 in modo che il pulsante SLEEP sul telecomando possa essere usato per disabilitare ed abilitare la navigazione del Boe-Bot in base agli altri tasti.

Soluzioni

Q1. "A riconoscere ed interpretare un segnale eletttronico."
Q2. Un bit.
Q3. Portando 2 alla potenza della posizione del bit.
Q4. Un bit è impostato quando è posto ad uno. Un bit è azzerato quando è posto a zero.
Q5. Il nome della variabile deve essere messo prima dell'operatore **`.BIT`**, e l'indirizzo del bit deve seguirlo. Non usare spazi. Per esempio, per usare l'operatore **`.BIT`** per riferirsi al bit-4 nella variabile **`remoteCode`**, usate **`remoteCode.BIT4`**.
Q6. Essi danno nomi significativi ai valori usati nel programma. Specificatamente, ai tasti non numerici del telecomando ci si può riferire con i loro nomi invece che con i loro valori numerici.
Q7. In base a quanto spiegato nel riquadro ?-box a pagina 44, significa che il pulsante non può più confondere con il flusso rapido di uno e zero inviati dall'incontro delle superfici metalliche della chiusura di un contatto. Sia la parte del circuito del pulsante che la parte del programma eliminano questi segnali imprevedibili.
Q8. L'altoparlante ci permette di sapere che la pressione di un tasto ha inviato il messaggio desiderato.

E1. 0, 1, 10, 11, 100, 101, 110, 111.
E2. 11000.
E3. $2^{12} = 4096$.
E4. 8 + 4 + 2 + 1 = 15.
E5. Dopo questi due comandi:

```
RCTIME IrDet, 0, irPulse
IF irPulse > 300 THEN remoteCode.BIT6 = 1
```

Aggiungete due ulteriori comandi per acquisire un altro impulso e quindi impostare/azzerare **`remoteCode.BIT7`**.

```
RCTIME IrDet, 0, irPulse
IF irPulse > 300 THEN remoteCode.BIT7 = 1
```

E6.

```
SELECT remoteCode
  CASE Power
    onOffState = ~ onOffState
ENDSELECT
```

P1. Questa soluzione richiede il circuito dell' altoparlante piezo che era stato aggiunto nell' Attività #4. Effettuate le seguenti modifiche al programma:

√ Aggiungete questa dichiarazione nella sezione Costanti:

```
RemoteSleep CON 54
```

√ Aggiungete una variabile di grandezza bit nella sezione Variabili. Chiamandola **`sleepOnOff`**.

```
' Boe-Bot control bit
sleepOnOff     VAR     Bit
```

√ Aggiungete questo comando alla sezione Inizializzazione che imposta **`sleepOnOff`** ad 1.

```
' -----[ Initialization ]-----------------------
sleepOnOff = 1
```

√ Aggiungete questo comando tra la chiamata alla subroutine **`Get_Ir_Remote_Code`** e la dichiarazione **`SELECT`**:

```
IF (sleepOnOff = 0) AND (remoteCode <> RemoteSleep) THEN
  remoteCode = 127
ENDIF
```

√ Aggiungete un comando **`CASE`** al blocco di codice **`SELECT…CASE`** come il seguente:

```
CASE RemoteSleep
  FREQOUT 4, 50, 4000
  PAUSE 50
  FREQOUT 4, 50, 4000
  PAUSE 300
  sleepOnOff = ~ sleepOnOff
```

√ Commentate i due comandi **`PULSOUT`** che seguono la dichiarazione **`CASE ELSE`**.

Capitolo 3: Ulteriori Applicazioni IR Remote

ESPANDERE I PROGRAMMI APPLICATIVI

Il punto chiave del codice riusabile è che vi permette di partire dal vostro lavoro precedente per realizzare applicazioni migliori e più potenti. Il lavoro precedente può essere codice che avete scritto voi, o può essere codice pubblicato che avete aggiunto o adattato. In questo capitolo, constaterete come le applicazioni su cui avete lavorato fino ad ora possono essere mescolate con altre applicazioni. Per esempio, il codice per Boe-Bot da *Robotica con il Boe-Bot* verrà aggiunto alla maschera del codice per l'applicazione Telecomando IR per effettuare una varietà di compiti.

Nell' Attività #1, mescolerete un programma per evitare gli oggetti autonomamente con gli IR in una maschera per la comunicazione remota. Il risultato sarà un Boe-Bot che avanza autonomamente e la cui velocità potrà essere da voi controllata con il telecomando. Nell' Attività #2, trasformerete il Boe-Bot in un robot multi funzione controllato a distanza. Con la pressione di un pulsante, potrete scegliere fra tre dei più popolari comportamenti del Boe-Bot: Boe-Bot controllato a distanza, Boe-Bot che esplora, e Boe-Bot inseguitore. Nell' Attività #3, progetterete un interprete per l'esecuzione di elenchi di istruzioni inviate dal telecomando IR. In pratica, sarete in grado di "programmare" il vostro Boe-Bot con il telecomando IR.

ATTIVITÀ #1: NAVIGAZIONE AUTONOMA CON IL CONTROLLO REMOTO DELLA VELOCITÀ

Questa attività svilupperà un esempio di navigazione autonoma con regolazioni da telecomando. Per la navigazione autonoma, il Boe-Bot esplorerà con gli infrarossi usando il Programma Esempio da *Robotica con il Boe-Bot* Capitolo 7, Attività #5. Questo programma esempio verrà quindi modificato in modo che possiate impostare il controllo della velocità dalla finestra di trasmissione del Terminale di Debug. Il programma della navigazione autonoma con IR con il controllo della velocità può essere mescolato con il programma per l'inserimento da tastiera. Dopo che questi due programmi sono stati combinati insieme, l'emissione di comandi per il controllo della velocità può essere effettuato con il telecomando IR invece che con il Terminale di Debug.

Uno degli errori più comuni dei progetti robotici è tentare di far funzionare tutte le componenti insieme in un colpo solo. Il risultato finale è di solito un buco troppo difficile da trovare. È meglio assicurarsi che ciascuna componente del progetto funzioni prima di integrarla in un sistema più grande. Con questo proposito in mente, questa attività è separata in quattro passi:

Passo 1 – Ricostruire, testare e collaudare il sistema di rivelazione IR seguendo i consigli di *Robotica con il Boe-Bot*, Capitolo 7, Attività #1 e #2.

Passo 2 – Testare la navigazione con la rilevazione IR di oggetti ed i programmi di interferenza IR che sono stati spiegati in *Robotica con il Boe-Bot*, Capitolo 7, Attività #5.

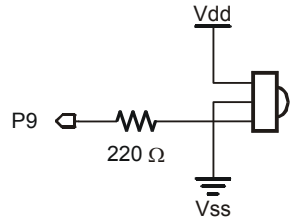
Passo 3 – Modificate il programma in modo che possiate controllare la velocità di navigazione con un comando **`DEBUGIN`**.

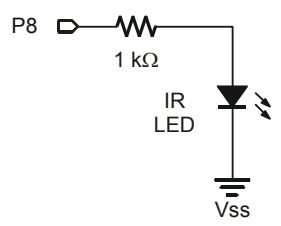
Passo 4 – Integrate la navigazione con la funzione di controllo della velocità nel programma applicativo di telecomando IR che supporta l'inserimento di grandi numeri con la tastiera.

Passo 1 - Ricostruire, testare e collaudare il Sistema di Rivelazione IR

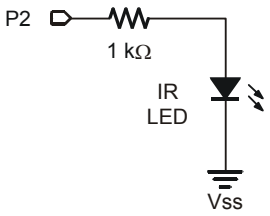
La Figura 3-1 mostra i circuiti per la rivelazione IR e gli indicatori per utente da *Robotica con il Boe-Bot*, Capitolo 7, Attività #2, e la Figura 3-2 mostra un modo di costruire questi circuiti con schemi di cablaggio.

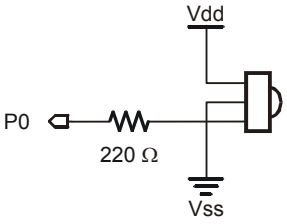
- √ Costruite i circuiti mostrati nella Figura 3-1 e nella Figura 3-2.
- √ Testate i circuiti seguendo le istruzioni da *Robotica con il Boe-Bot*, Capitolo 7, Attività #2. Userete sia TestIrPairsAndIndicators.bs2 che IrInterferenceSniffer.bs2.





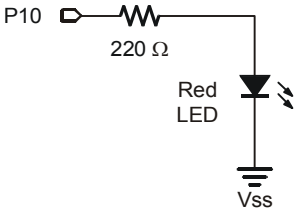
Coppia IR di Sinistra



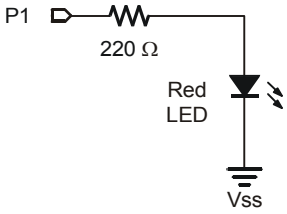


Coppia IR di Destra

Figura 3-1 Circuiti della Rivelazione IR per il Test e la Navigazione



Indicatore LED per la Coppia IR di Sinistra



Indicatore LED per la Coppia IR di Destra

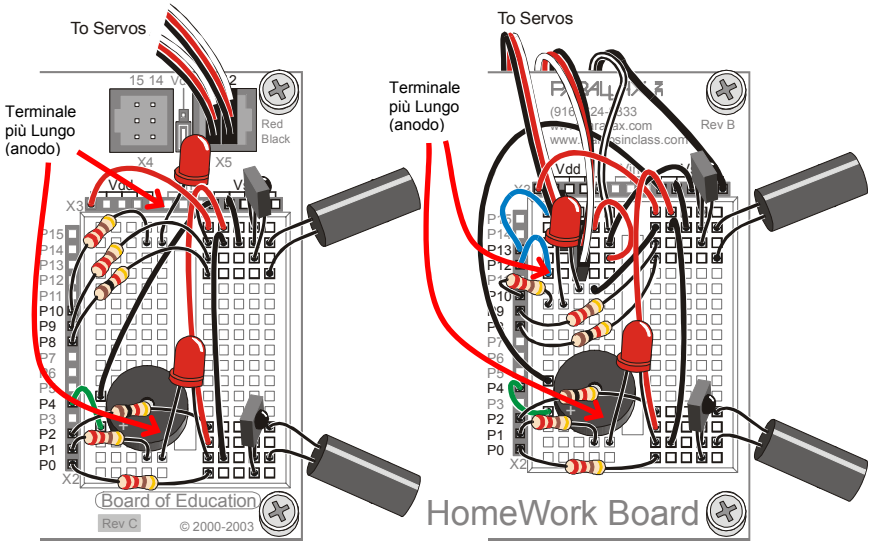


Figura 3-2
Schemi di Cablaggio dei Circuiti di Rivelazione IR Teste e Navigazione

Passo 2 - Testare il programma di Navigazione con la Rilevazione IR di Oggetti

Il Capitolo 7, Attività #5 di *Robotica con il Boe-Bot* presenta una versione ad alte prestazioni della navigazione con gli IR. Questo programma controlla per la presenza di un oggetto prima dell'invio di ogni impulso ai servo (che avviene circa 40 volte al secondo), quindi il Boe-Bot è molto rapido a reagire quando rileva ostacoli. In seguito modificherete questo programma, quindi andate avanti e rinfrescate la conoscenza del suo funzionamento. È anche importante assicurarsi che il programma originale funzioni con il vostro circuito prima di effettuare altre modifiche.

- √ Rivedete il programma FastIrRoaming.bs2 e la spiegazione di come funziona in *Robotica con il Boe-Bot*, Capitolo 7, Attività #5.
- √ Aprite (o ridigitate) e provate FastIrRoaming.bs2 seguendo le istruzioni di questa Attività.

Passo 3 - Modificate il Programma in modo che possiate Controllare la Velocità di Navigazione

Le variabili possono essere usate per controllare una quantità di comportamenti del Boe-Bot. tra le altre cose, queste variabili possono indicare al Boe-Bot quanto andare veloce

(questo passo) e quale compito eseguire (il prossimo passo). Alcuni esempi di condizioni che possono essere usati per cambiare queste variabili sono:

- Ingresso da Sensori
- Messaggi dal Terminale di Debug
- Messaggi da un telecomando IR

Nel prossimo programma esempio, viene aggiunta una variabile chiamata **speed** in modo che la velocità dei servo possa essere impostata con un comando **DEBUGIN**. Inserendo un valore tra 0 e 100, potrete far navigare il Boe-Bot ad una qualsiasi velocità tra lo 0 ed il 100% della sua velocità massima.

La dichiarazione **IF…THEN** da FastIrRoaming.bs2 in effetti necessita una qualche ristrutturazione prima che possa controllare la velocità. Invece di usare i valori 650 ed 850 per la massima velocità, i valori di **pulseLeft** e di **pulseRight** dovrebbe essere determinare dichiarando una variabile **speed** da aggiungere o sottrarre a 750. La Table 3-1 mostra come appare la dichiarazione originale **IF…THEN** da FastIrRoaming.bs2 confontata con la versione modificata della dichiarazione **IF…THEN** dal prossimo programma esempio, IrRoamingWithSpeedControl.bs2.

Table 3-1: Codice di Navigazione con/senza Controllo della Velocità

Senza il Controllo della Velocità	Con il Controllo della Velocità

`IF (irDetectLeft = 0) AND ••• THEN`
`  pulseLeft = 650`
`  pulseRight = 850`
`ELSEIF (irDetectLeft = 0) THEN`
`  pulseLeft = 850`
`  pulseRight = 850`
`ELSEIF (irDetectRight = 0) THEN`
`  pulseLeft = 650`
`  pulseRight = 650`
`ELSE`
`  pulseLeft = 850`
`  pulseRight = 650`
`ENDIF`
	`IF (irDetectLeft = 0) AND ••• THEN`
	`  pulseLeft  =  750 - speed`
	`  pulseRight =  750 + speed`
	`ELSEIF (irDetectLeft = 0) THEN`
	`  pulseLeft  = 750 + speed`
	`  pulseRight = 750 + speed`
	`ELSEIF (irDetectRight = 0) THEN`
	`  pulseLeft = 750 - speed`
	`  pulseRight = 750 - speed`
	`ELSE`
	`  pulseLeft = 750 + speed`
	`  pulseRight = 750 - speed`
	`ENDIF`

Sulla parte "Con il Controllo della Velocità" della tabella, i comandi **PULSOUT** per i servo sinistro e destro usano le variabili **pulseLeft** e **pulseRight** per gli argomenti

Duration. Il comando **pulseLeft = 650** viene sostituito con **pulseLeft = 750 – speed**. Quando **speed** è un valore piccolo, **pulseLeft** è vicino a 750, e la ruota sinistra del Boe-Bot gira molto lentamente in senso orario. Quando la variabile **speed** si avvicina a 100, la ruota sinistra del Boe-Bot si avvicina alla sua velocità massima in senso orario. Ora, osservando come è stato sostituito **pulseRight = 850** con **pulseRight = 750 + speed**. Quando **speed** è piccolo, la ruota destra gira molto lentamente in senso antiorario, e quando **speed** è prossima a 100, essa gira in senso antiorario alla massima velocità.

Programma Esempio: IrRoamingWithSpeedControl.bs2

Questo programma esempio è una versione modificata di FastIrRoaming.bs2 da *Robotica con il Boe-Bot*, Capitolo 7, Attività #5. i commenti nel listato del programma vi mostreranno quali righe dovrete aggiungere o cambiare per completare le modifiche.

Prima di lanciare IrRoamingWithSpeedControl.bs2, il Terminale di Debug vi richiederà di digitare una velocità tra 0 e 100. Dopo che avete inserito la velocità desiderata, il Boe-Bot navigherà a quella percentuale della velocità massima.

- √ Digitate e lanciate IrRoamingWithSpeedControl.bs2.
- √ Quando lanciate il programma, il Terminale di Debug vi richiederà la percentuale della velocità massima a cui volete che il vostro Boe-Bot navighi. Digitate la velocità desiderata nella Finestra di Trasmissione del Terminale di Debug.

Per un ripasso su come usare la Finestra di Trasmissione del Terminale di Debug, vedere la Figura 1-12 a pagina 22.

- √ Provate a lanciare il programma alcune volte, ogni volta selezionando una differente percentuale della velocità massima.

```
' IR Remote for the Boe-Bot - IrRoamingWithSpeedControl.bs2
' Higher performance IR object detection assisted navigation.
' The "<-- Add" comments indicate new commands lines of code.
' The "<-- Change" comments indicate lines of code that should be changed.

' {$STAMP BS2}
' {$PBASIC 2.5}

irDetectLeft   VAR     Bit                   ' Variable Declarations
```

```
irDetectRight  VAR     Bit
pulseLeft      VAR     Word
pulseRight     VAR     Word
speed          VAR     Byte             ' <-- Add

FREQOUT 4, 2000, 3000                       ' Signal program start/reset.

DEBUG CLS, "Enter percent of", CR,      ' <-- Add
           "full speed (0 TO 100): "    ' <-- Add
DEBUGIN DEC speed                       ' <-- Add
DEBUG "Main routine running..."         ' <-- Add

DO                                          ' Main Routine

  FREQOUT 8, 1, 38500                       ' Check IR Detectors
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  irDetectRight = IN0
                                            ' Decide how to navigate.
  IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
    pulseLeft  =  750 - speed           ' <-- Change
    pulseRight =  750 + speed           ' <-- Change
  ELSEIF (irDetectLeft = 0) THEN
    pulseLeft  = 750 + speed            ' <-- Change
    pulseRight = 750 + speed            ' <-- Change
  ELSEIF (irDetectRight = 0) THEN
    pulseLeft = 750 - speed             ' <-- Change
    pulseRight = 750 - speed            ' <-- Change
  ELSE
    pulseLeft = 750 + speed             ' <-- Change
    pulseRight = 750 - speed            ' <-- Change
  ENDIF

  PULSOUT 13,pulseLeft                      ' Apply the pulse.
  PULSOUT 12,pulseRight
  PAUSE 15

LOOP                                        ' Repeat main routine
```

Come Funziona IrRoamingWithSpeedControl.bs2

Nella sezione Dichiarazioni, viene aggiunta una variabile di controllo della velocità. La variabile è saggiamente chiamata **speed**.

```
speed          VAR     Byte             ' <-- Add
```

un comando **DEBUG** vi richiede di digitare la percentuale della velocità massima a cui volete che il vostro Boe-Bot navighi.

```
DEBUG CLS, "Enter percent of", CR,      ' <-- Add
           "full speed (0 TO 100): "    ' <-- Add
```

Un comando **DEBUGIN** memorizza il valore che avete digitato nella variabile **speed**. Se il valore di **speed** viene impostato a 100 dall'utente, il Boe-Bot navigherà alla sua massima velocità. Quando viene inserito un valore minore di 100, il Boe-Bot navigherà a quella percentuale della velocità massima.

```
DEBUGIN DEC speed                       ' <-- Add
```

Il valore percentuale che avete digitato determina la percentuale della larghezza di impulso per la massima velocità. Questa non è la stessa della velocità attuale. Per un aiuto nella predizione della velocità attuale, consultate le curve di trasferimento studiate in *Robotica con il Boe-Bot,* Capitolo 3, Attività #4.

Un messaggio **DEBUG** indica che la routine principale stà funzionando.

```
DEBUG "Main routine running..."         ' <-- Add
```

Questa dichiarazione **IF…THEN** è stata studiata subito prima del programma esempio. Essa imposta i valori delle variabili **pulseLeft** e **pulseRight**. In base alla direzione in cui dovrà girare ciascun servo, il valore **speed** viene quindi sommato o sottratto da 750 per impostare la velocità dei servo.

```
IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
  pulseLeft  =  750 - speed             ' <-- Change
  pulseRight =  750 + speed             ' <-- Change
ELSEIF (irDetectLeft = 0) THEN
  pulseLeft  = 750 + speed              ' <-- Change
  pulseRight = 750 + speed              ' <-- Change
ELSEIF (irDetectRight = 0) THEN
  pulseLeft = 750 - speed               ' <-- Change
  pulseRight = 750 - speed              ' <-- Change
ELSE
  pulseLeft = 750 + speed               ' <-- Change
  pulseRight = 750 - speed              ' <-- Change
ENDIF
```

Dopo che i valori delle variabili **pulseLeft** e **pulseRight** sono stati impostati dalla dichiarazione **IF…THEN**, queste variabili sono usate per impostare gli argomenti ***Duration*** dei comandi **PULSOUT**. Questi comandi **PULSOUT** inviano gli impulsi ai servo sinistro e destro, e le durate degli impulsi determinano la direzione e la velocità a cui girano le ruote del Boe-Bot.

```
PULSOUT 13,pulseLeft                    ' Apply the pulse.
PULSOUT 12,pulseRight
PAUSE 15
```


Il Vostro Turno – Risparmiare Spazio Variabile

Potete risparmiare lo spazio RAM di due variabili di grandezza word usando i comandi **PULSOUT** all'interno della dichiarazione **IF…THEN**. Ecco come:

- √ Salvate il vostro programma funzionante con un altro nome (come ad esempio IrRoamingWithSpeedControlYourTurn.bs2).
- √ Clickate Run e selezionate Memory Map (o CTRL-M o l'icona della Mappa della Memoria sulla barra degli utensili).
- √ Controllate la Mappa della RAM per vedere quanta RAM state usando. La codifica dei colori dovrebbe mostrare che REG 0 e 1 sono variabili per memorizzare word. Dovrebbe anche mostrare che REG 2 ha una variabile byte e due variabili bit, come mostrato sulla sinistra della Figura 3-3.
- √ Commentate le dichiarazioni delle variabili **pulseLeft** e **pulseRight**.
- √ Sostituite tutte le istanze di **pulseLeft =** con **PULSOUT 13**, .
  Fate attenzione quì, il risultato della vostra prima sostituzione dovrebbe essere:

  ```
  PULSOUT 13, 750 - speed
  ```

- √ Sostituite tutte e 4 le istanze di **pulseRight =** con **PULSOUT 12**, .
  Di nuovo, il risultato delle vostre quattro sostituzioni dovrebbero essere:

  ```
  PULSOUT 12, 750 + speed
  ```

- √ Decommentate i due comandi **PULSOUT** che nel programma vengono dopo il comando **ENDIF** (inserendo un apostrofo alla sinistra di ciascuno dei due comandi).
- √ Salvate, lanciate, e provate il programma modificato. Se necessario risolvete i problemi.
- √ Riaprite la Mappa della Memoria, e verificate che in **REG 0** siano allocate solamente una variabile byte e due variabili bit, come mostrato sulla destra nella Figura 3-3.

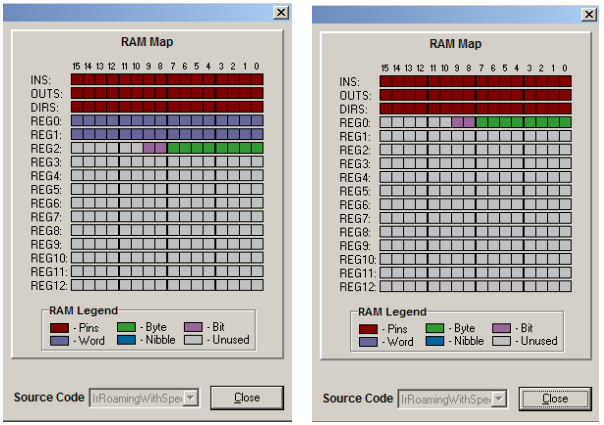


Figura 3-3
La porzione della Mappa RAM nella Mappa della Memoria

IrRoamingWithSpeedControl.bs2 (sinistra)

Versione Modificata ne Il Vostro Turno (destra)

Passo 4 - Integrare la Navigazione con il Controllo della Velocità nella Maschera IR

Combinando IrRoamingWithSpeedControl.bs2 con IrRemoteKeypad.bs2, potrete usare il telecomando IR per regolare la velocità del Boe-Bot mentre naviga. Ecco come esso funzionerà:

- Premendo un pulsante qualsiasi, interromperete la navigazione del Boe-Bot.
- Quindi, userete la tastiera numerica del telecomando per digitare la percentuale della nuova velocità.
- Per far riprendere la navigazione al Boe-Bot con la nuova velocità, premete il pulsante ENTER.

IrRoamingWithSpeedControl.bs2 già usa gli infrarossi per rilevare gli ostacoli tra ogni coppia di impulsi per i servo. La chiave per rilevare messaggi in arrivo dal telecomando è controllare i rivelatori IR prima di usare il comando **FREQOUT** per cercare eventuali ostacoli. Se il programma controlla per trovare se sono rilevati IR prima di controllare per eventuali ostacoli, può facilmente rilevare un messaggio in arrivo dal telecomando. Ecco i passi che dovrebbero essere eseguiti all'interno del ciclo **DO…LOOP** della routine principale:

- Prima di controllare per gli ostacoli, con i LED IR del Boe-Bot, controllare il piedino di uscita del rivelatore IR per vedere se un segnale stà giungendo dal telecomando.

  - Se il rivelatore IR stà emettendo un segnale basso significa che stà arrivando un messaggio infrarosso. Chiamare la subroutine **`Process_Ir_Message`**.
  - Se il rivelatore IR stà emettendo un segnale alto, andare al prossimo compito che è la rilevazione di ostacoli.
- Controllare i rivelatori IR.
- Usate **`IF…THEN`** per controllare la direzione e la velocità dei servo.

Progamma Esempio: RoamingWithRemoteSpeedControl.bs2

Seguite questi passi per scrivere il programma:

√ Aprire IrRemoteKeypad.bs2.
√ Salvarne una copia con il nome di RoamingWithRemoteSpeedControl.bs2.

Per trasferire delle sezioni di codice dalla vostra versione di IrRoamingWithSpeedControlYourTurn.bs2 in RoamingWithRemoteSpeedControl.bs2 potete usare copia ed incolla.

√ Aggiungete queste dichiarazioni alla sezione variabili:

```
' Boe-Bot control variables.

irDetectLeft   VAR     Bit                  ' IR detector bit storage
irDetectRight  VAR     Bit
speed          VAR     Byte                 ' Speed control variable
```

√ Aggiungete una sezione di Inizializzazione con questo codice di avvio per il Boe-Bot subito prima della sezione della routine principale.

```
' -----[ Initialization ]-------------------------------------------

' Boe-Bot initialization.

  DEBUG "Starting...", CR, CR           ' Signal program start/reset.
  FREQOUT 4, 2000, 3000
  speed = 0                             ' Initial speed is zero.
```

√ Sostituite il ciclo **`DO…LOOP`** nella routine principale con questo:

```
DO                                      ' Main Routine.

  IF (IN9 = 0) OR (speed = 0) THEN      ' Check for IR message.
    FREQOUT Speaker, 100, 3500          ' Signal remote message
```

```
      PAUSE 100                                 ' detected.

      FREQOUT Speaker, 100, 3500
      PAUSE 200

      DEBUG "Speed range: 0 to 100", CR, CR,
            "Type speed on remote", CR,
            "keypad, then press", CR,
            "ENTER", CR, CR

      GOSUB Get_Multi_Digit_Value               ' Get new speed.
      speed = value                             ' Set new speed.

      DEBUG ? speed, CR

      DEBUG "Running...", CR,
            "Press any key to", CR,
            "interrupt roaming", CR, CR

      PAUSE 250                                 ' Pause for debounce.
    ENDIF

    FREQOUT 8, 1, 38500                         ' Check IR Detectors.
    irDetectLeft = IN9
    FREQOUT 2, 1, 38500
    irDetectRight = IN0
                                                ' Decide how to navigate.
    IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
      PULSOUT 13,  750 - speed                  ' Backward
      PULSOUT 12,  750 + speed
    ELSEIF (irDetectLeft = 0) THEN
      PULSOUT 13, 750 + speed                   ' Rotate right
      PULSOUT 12, 750 + speed
    ELSEIF (irDetectRight = 0) THEN
      PULSOUT 13, 750 - speed                   ' Rotate left
      PULSOUT 12, 750 - speed
    ELSE
      PULSOUT 13, 750 + speed                   ' Forward
      PULSOUT 12, 750 - speed
    ENDIF

    PAUSE 15                                    ' Between servo pulses.

  LOOP                                          ' Repeat main routine.
```

√ Salvate e lanciate il vosto programma modificato.

√ Per interrompere la navigazione del Boe-Bot, puntategli il telecomando e premete un pulsante qualsiasi. Il Boe-Bot emetterà due beep per indicare che ha

ricevuto il segnale. Assicuratevi di rilasciare il pulsante che avete premuto non appena udite i beep.

√ Usate la tastiera numerica del telecomando per digitare il nuovo valore percentuale dlla velocità (da 0 a 100).

√ Premete il pulsante ENTER per far riprendere al Boe-Bot la sua navigazione con la nuova velocità.

Come Funziona RoamingWithRemoteSpeedControl.bs2

Normalmente, durante la rivelazione IR, un comando **FREQOUT** fà emettere al LED IR dei lampi di infrarossi alla frequenza di 38.5 kHz. Se questi infrarossi vengono riflessi da un oggetto sul percorso del Boe-Bot, esso verrà rilevato dal rivelatore di IR. Ricordate che questo valore deve essere memorizzato in una variabile bit immediatamente prima del comando **FREQOUT**. Un istante dopo, l'uscita del rivelatore di IR ritorna in stato alto indicando che non si rilevano infrarossi.

Durante la navigazione senza che ci siano messaggi IR dal telecomando, il ciclo **DO…LOOP** della routine principale viene eseguito in continuazione, per controllare con gli IR se ci sono ostacoli e per controllare i servo. Il modo di intercettare un messaggio IR remoto è di controllare lo stato delle uscite dei rivelatori IR prima di emettere infrarossi con i LED IR. Il momento adatto per far questo è all'inizio del ciclo **DO…LOOP** della routine principale, perchè questo viene ad essere prima della pausa di 15 ms alla fine del ciclo **DO…LOOP**.

La dichiarazione **IF (IN9 = 0) OR (speed = 0) THEN...** effettua il controllo effettivo di un messaggio IR remoto. Esso controlla anche se la variabile **speed** è stata impostata o meno. La prima volta che il programma viene lanciato, il valore di **speed** è inizializzato a zero. La dichiarazione **IF…THEN** lo rileva ed il blocco di codice al suo internovi inviata ad inserire una velocità. La dichiarazione **IF…THEN** rileva anche quando un telecomando stà inviando un messaggio. Quando un messaggio infrarosso viene emesso, **IN9 = 0**. Il blocco di codice inizia suonando due volte per farvi sapere che potete rilasciare il pulsante sul telecomando. Quindi, chiama la subroutine **Get_Multi_Digit_Value**, che è quella che acquisisce i tasti premuti per impostare la nuova velocità, seguiti dal tasto ENTER per far ripartire il Boe-Bot. quando un messaggio infrarosso non viene emesso, **IN9 = 1**, ed il blocco di codice tra **IF** ed **ENDIF** non viene eseguito. Il risultato è che il Boe-Bot continua la sua navigazione.

```
IF (IN9 = 0) OR (speed = 0) THEN          ' Check for IR message.
  FREQOUT Speaker, 100, 3500              ' Signal remote message
```

```
  PAUSE 100                                 ' detected.
     •
     •
     •
  GOSUB Get_Multi_Digit_Value               ' Get new speed.
  speed = value                             ' Set new speed.
     •
     •
     •
  PAUSE 250                                 ' Pause for debounce.
ENDIF
```

Il Vostro Turno – Discriminare il tasto POWER

Al momento, potete interrompere la navigazione del Boe-Bot premendo qualsiasi tasto TV sul telecomando. Certamente, questo non funzionerà se premete tasti sul telecomando che non sono assegnati al controllo della TV. Assumete di voler usare, per interrompere la navigazione del Boe-Bot, solamente il tasto POWER.

Potete modificare il codice in modo che la subroutine **`Get_Multi_Digit_Value`** sia richiamata solamente quando viene premuto il tasto POWER annidando tutto il codice che interpreta la nuova velocità emessa dal telecomando in una seconda dichiarazione **`IF…THEN`**. Prima di questa dichiarazione **`IF…THEN`**, viene chiamata la subroutine **`Get_Ir_Remote_Code`**. Ricordate che questa subroutine memorizza il valore dei tasti del telecomando ricevuti nella variabile **`remoteCode`**. In questo modo, questa seconda, interna dichiarazione **`IF…THEN`** può controllare **`remoteCode`**. Se è uguale alla costante **`Power`**, allora aggiorna la variabile **`speed`**; altrimenti, emette un codice di errore con l'altoparlante piezo.

- √ Salvate RoamingWithRemoteSpeedControl.bs2 con il nome di RoamingWithRemoteSpeedControlYourTurn.bs2.
- √ Sostituite questa dichiarazione **`IF…THEN`**:

```
IF (IN9 = 0) OR (speed = 0) THEN            ' Check for IR message.
  FREQOUT Speaker, 100, 3500                ' Signal remote message
  PAUSE 100                                 ' detected.
     •
     •
     •
  GOSUB Get_Multi_Digit_Value               ' Get new speed.
  speed = value                             ' Set new speed.
     •
     •
     •
```

```
  PAUSE 250                                 ' Pause for debounce.
ENDIF
```

Con questa:

```
IF (IN9 = 0) OR (speed = 0) THEN            ' Check for IR message.
  DEBUG "Press POWER to set", CR,
        "speed...", CR, CR
  GOSUB Get_Ir_Remote_Code
  IF (remoteCode = Power) OR (speed = 0) THEN
    FREQOUT Speaker, 100, 3500              ' Signal remote message
    PAUSE 100                               ' detected.

    FREQOUT Speaker, 100, 3500
    PAUSE 200

    DEBUG "Speed range: 0 to 100", CR, CR,
          "Type speed on remote", CR,
          "keypad, then press", CR,
          "ENTER", CR, CR

    GOSUB Get_Multi_Digit_Value             ' Get new speed.
    speed = value                           ' Set new speed.

    DEBUG ? speed, CR

    DEBUG "Running...", CR, CR

    PAUSE 250                               ' Pause for debounce.
  ELSE
    GOSUB Beep_Error
  ENDIF
ENDIF
```

√ Salvate i cambiamenti che avete fatto, quindi lanciate e testate il programma.

ATTIVITÀ #2: BOE-BOT MULTI-FUNZIONI CON SELEZIONE REMOTA

Con la pressione di un tasto del telecomando, potrete selezionare una fra tre differenti funzioni del Boe-Bot:

- Boe-Bot controllato dal telecomando
- Boe-Bot a navigazione autonoma
- Boe-Bot inseguitore

Selezionare tra Routine Principali

Il prossimo programma esempio inizia con 7BitRemoteBoeBot.bs2 da questo testo. La routine principale da questo programma e la routine principale da FastIrRoaming.bs2 e da FollowingBoeBot.bs2 (da *Robotica con il Boe-Bot*) possono tutte essere mescolate in dichiarazioni **`CASE`**. La routine principale di questo nuovo programma può quindi usare una grande, singola, dichiarazione **`SELECT…CASE`** che usa una variabile chiamata **`operation`** per selezionare quale routine eseguire. Il risultato è un Boe-Bot con tre funzioni selezionabili da telecomando IR.

Ricordate che le direttive **`PIN`**, le dichiarazioni di costanti e di variabili, e le subroutine devono anch'esse essere importate da FastIrRoaming.bs2 e da FollowingBoeBot.bs2.

Far funzionare tutte le routine insieme necessita di alcuni aggiustamenti. Ci deve essere un qualche modo per interrompere il compito corrente del Boe-Bot, che sia la navigazione, il controllo remoto o l'inseguimento, in modo che gli possiate comunicare di eseguire un compito differente. Dal momento che i tasti numerici, di canale e di volume sono già stati impegnati per uno dei compiti, per interrompere ancora il compito corrente del Boe-Bot, useremo il tasto POWER.

Ecco come la routine principale del programma funzionerà:

```
DO
  SELECT operation

    ' If operation = 1, execute a modified version of
    ' 7BitRemoteBoeBot.bs2 that also allows
    ' you to change the operation variable with the POWER key.
    CASE 1

       ' Modified main routine from 7BitRemoteBoeBot.bs2
       ' goes here.
            •
            •
            •

    ' If operation = 2, execute modified FastIrRoaming.bs2.
    CASE 2

       ' Modified main routine from FastIrRoaming.bs2 goes here.
            •
            •
            •

    ' If operation is 3, execute the FollowingBoeBot.bs2.
    CASE 3

       ' Modified main routine from FollowingBoeBot.bs2 goes here.
            •
            •
            •

  ENDSELECT                                  ' End SELECT operation

LOOP
```

Per prima cosa assembliamo e testiamo il programma, quindi daremo uno sguardo ravvicinato al suo funzionamento.

Progamma Esempio: IrMultiBot.bs2

Questo Programma Esempio inizia come il 7BitRemoteBoeBot.bs2 normale. Quindi, potrete usare i tasti numerici, del canale e del volume per guidare la navigazione del Boe-Bot. Potrete effettuare la navigazione autonoma del vostro Boe-Bot premendo il tasto POWER, quindi il tasto 2. Potrete far inseguire al Boe-Bot degli oggetti premendo il tasto POWER, quindi il tasto 3. per tornare alla navigazione controllata dalla tastiera del telecomando, premere il tasto POWER, quindi il tasto 1.

Ecco come assemblare il programma. Primo, ciascuno dei tre programmi che userete per assemblare questo programma più grande dovranno essere funzionanti e collaudati. 7BitRemoteBoeBot.bs2 è stato sviluppato in questo testo, al Capitolo 2, Attività #3. gli altri due programmi sono stati sviluppati in *Robotica con il Boe-Bot*. FastIrRoaming.bs2 è stato sviluppato nel Capitolo 7, Attività #5, e FollowingBoeBot.bs2 è stato sviluppato nel Capitolo 8, Attività #2.

- √ Caricate, lanciate e testate 7BitRemoteBoeBot.bs2.
- √ Caricate, lanciate e testate FastIrRoaming.bs2.
- √ Caricate, lanciate e testate FollowingBoeBot.bs2.

Quando avete lanciato e verificato che ciascuno dei tre programmi funziona correttamente da solo, sarete pronti ad iniziare la loro integrazione.

- √ Salvate una copia di 7BitRemoteBoeBot.bs2 con il nome di IrMultiBot.bs2.
- √ Aggiornate il titolo e il commento descrittivo nella sezione Titolo.

```
' -----[ Title ]-------------------------------------------------------
' IR Remote for the Boe-Bot - IrMultiBot.bs2
' Select one of three Boe-Bot behaviors with the IR remote 1-3 keys.

' Press POWER key to interrupt the Boe-Bot's operation.
' Then, press one of these digit keys to select a new mode:

' 1 - Control Boe-Bot with 1-9 keys and/or CH+/- and VOL+/- keys.
' 2 - Roam and avoid objects.
' 3 - Follow objects.

' Note: Startup default is mode 1.

' {$STAMP BS2}
' {$PBASIC 2.5}
```

Troverete utile la direttiva **`Speaker PIN`**.

- √ Aggiungete questa dichiarazione alle Definizioni I/O:

```
Speaker        PIN     4
```

Quindi, vi serviranno le costanti, le variabili ed un paio di subroutine da FollowingBoeBot.bs2. Dal momento che FollowingBoeBot.bs2 è assemblato su

FastIrRoaming.bs2, non vi serviranno costanti , variabili o subroutine aggiuntive da FastIrRoaming.bs2.

√ Aggiungete queste dichiarazioni di costanti nella sezione Costanti del programma:

```
' Boe-Bot proportional control constants (from FollowingBoeBot.bs2).

Kpl             CON     -35
Kpr             CON     35
SetPoint        CON     2
CenterPulse     CON     750
```

√ Copiate queste dichiarazioni di variabili da FollowingBoeBot.bs2, ed incollatele nella sezione Variabili di IrMultiBot.bs2.

```
' Boe-Bot navigation variables (from FollowingBoeBot.bs2).

freqSelect      VAR     Nib
irFrequency     VAR     Word
irDetectLeft    VAR     Bit
irDetectRight   VAR     Bit
distanceLeft    VAR     Nib
distanceRight   VAR     Nib
pulseLeft       VAR     Word
pulseRight      VAR     Word
```

√ Aggiungete due ulteriori dichiarazioni di variabili per le funzioni di IrMultiBot.bs2.

```
' IrMultiBot.bs2 variables.

counter         VAR     Nib                     ' <--- New
operation       VAR     Nib                     ' <--- New
```

√ Alla fine del programma aggiungete queste due subroutine:

```
' -----[ Subroutine – Get_IR_Distances ]------------------------------

Get_Ir_Distances:
  distanceLeft = 0
  distanceRight = 0
  FOR freqSelect = 0 TO 4
    LOOKUP freqSelect,[37500,38250,39500,40500,41500], irFrequency

    FREQOUT 8,1,irFrequency
    irDetectLeft = IN9
    distanceLeft = distanceLeft + irDetectLeft
```

```
    FREQOUT 2,1,irFrequency
    irDetectRight = IN0
    distanceRight = distanceRight + irDetectRight
  NEXT
  RETURN

' -----[ Subroutine - Send_Pulse ]-----------------------------------

Send_Pulse:
  PULSOUT 13,pulseLeft
  PULSOUT 12,pulseRight
  PAUSE 5
  RETURN
```

√ Sostituite la routine di inizializzazione con quella mostrata di seguito:

```
' -----[ Initialiazation ]------------------------------------------

DEBUG "Press POWER to select", CR,
      "mode of operation:", CR, CR,
      "1 – Remote control 1-9 & CH/VOL", CR,
      "2 – Autonomous IR roaming", CR,
      "3 – Object find and follow", CR, CR

FREQOUT Speaker, 2000, 3000

operation = 1                                ' Initialize to remote.
```

√ Modificate la sezione della routine principale per renderla simile a quella seguente. Sebbene possiate abbondantemente prendere in prestito (copiare/tagliare ed incollare) dalle routine principali dei programmi che avete caricato e testato, dovrete ancora fare degli aggiustamenti a ciascuna delle dichiarazioni **`CASE`**. Le righe modificate o nuove, avranno commenti come: **`<--- Modificato`** o **`<--- Nuovo`**.

```
DO

  SELECT operation                           ' <--- Nuovo

    ' If operation = 1, execute a modified version of
    ' 7BitRemoteBoeBot.bs2 that also allows
    ' you to change the operation variable with the POWER key.

    CASE 1                                   ' <--- Nuovo

      ' Modified main routine from 7BitRemoteBoeBot.bs2
      ' goes here.
```

```
  GOSUB Get_Ir_Remote_Code

  ' Check for POWER button.  If yes, get remote code; otherwise,
  ' send PULSOUT durations for the various maneuvers based on
  ' the value of the remoteCode variable.

  SELECT remoteCode
    CASE Power                                  ' <--- Nuovo
      FREQOUT Speaker, 100, 3500                ' <--- Nuovo
      PAUSE 100                                 ' <--- Nuovo
      FREQOUT Speaker, 100, 3500                ' <--- Nuovo
      PAUSE 200                                 ' <--- Nuovo
      DEBUG "Select operation mode...", CR      ' <--- Nuovo
      GOSUB Get_Ir_Remote_Code                  ' <--- Nuovo
      operation = remoteCode                    ' <--- Nuovo
      DEBUG ? operation, CR,                    ' <--- Nuovo
            "Running...", CR                    ' <--- Nuovo
      FREQOUT Speaker, 100, 3500                ' <--- Nuovo
      PAUSE 100                                 ' <--- Nuovo
      FREQOUT Speaker, 100, 3500                ' <--- Nuovo
      PAUSE 200                                 ' <--- Nuovo
    CASE 2, ChUp                                ' Forward
      PULSOUT 13, 850
      PULSOUT 12, 650
    CASE 4, VolDn                           ' Rotate Left
      PULSOUT 13, 650
      PULSOUT 12, 650
    CASE 6, VolUp                           ' Rotate Right
      PULSOUT 13, 850
      PULSOUT 12, 850
    CASE 8, ChDn                            ' Backward
      PULSOUT 13, 650
      PULSOUT 12, 850
    CASE 1                                  ' Pivot Fwd-left
      PULSOUT 13, 750
      PULSOUT 12, 650
    CASE 3                                  ' Pivot Fwd-right
      PULSOUT 13, 850
      PULSOUT 12, 750
    CASE 7                                  ' Pivot Back-left
      PULSOUT 13, 750
      PULSOUT 12, 850
    CASE 9                                  ' Pivot Back-right
      PULSOUT 13, 650
      PULSOUT 12, 750
    CASE ELSE                               ' Hold Position
      PULSOUT 13, 750
      PULSOUT 12, 750
  ENDSELECT

' If operation = 2, execute modified FastIrRoaming.bs2.
```

```
      CASE 2                                    ' <--- Nuovo

        ' Modified main routine from FastIrRoaming.bs2 goes here.

        IF IN9 = 0 THEN operation = 1           ' <--- Nuovo.

        FREQOUT 8, 1, 38500                     ' Check IR Detectors
        irDetectLeft = IN9
        FREQOUT 2, 1, 38500
        irDetectRight = IN0
                                                 ' Decide how to navigate.
        IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
          pulseLeft = 650
          pulseRight = 850
        ELSEIF (irDetectLeft = 0) THEN
          pulseLeft = 850
          pulseRight = 850
        ELSEIF (irDetectRight = 0) THEN
          pulseLeft = 650
          pulseRight = 650
        ELSE
          pulseLeft = 850
          pulseRight = 650
        ENDIF

        GOSUB Send_Pulse                        ' <--- Modificato.

      ' If operation is 3, execute the following Boe-Bot routine.
      CASE 3                                    ' <--- Nuovo

        IF IN9 = 0 THEN operation = 1           ' <--- Nuovo

        GOSUB Get_Ir_Distances

        ' Calculate proportional output.

        pulseLeft =  SetPoint - distanceLeft  * Kpl + CenterPulse
        pulseRight = SetPoint - distanceRight * Kpr + CenterPulse

        GOSUB Send_Pulse

    ENDSELECT                                   ' <--- Nuovo
                                                '(End SELECT operation)
  LOOP                                          ' Repeat Main Routine.
```

Il programma completato è mostrato sotto nel caso dobbiate controllarlo per la ricerca di errori.

√ Assemblate, salvate, e lanciate IrMultiBot.bs2.

- √ Testate i tasti CH+/-, VOL+/-, e numerici e verificate che funzionino correttamente.
- √ Premere/rilasciare il tasto POWER. Il Boe-Bot dovrebbe suonare due volte.
- √ Premere/rilasciare il tasto 2.
- √ Il Boe-Bot dovrebbe suonare due volte, quindi iniziare la navigazione autonoma e ad evitare gli ostacoli.
- √ Premere/rilasciare il tasto POWER. Il Boe-Bot dovrebbe suonare due volte.
- √ Premere/rilasciare il tasto 3.
- √ Il Boe-Bot dovrebbe ora navigare in modalità inseguitore. Collaudarlo per assicurarvi che si agganci e segua un oggetto.
- √ Premere/rilasciare il tasto POWER. Il Boe-Bot dovrebbe suonare due volte.
- √ Premere/rilasciare il tasto 1.
- √ Questo dovrebbe far tornare il Boe-Bot in modalità controllo con telecomando (tasti CH/VOL +/- e numerici).

```
' -----[ Title ]-------------------------------------------------------------
' IR Remote for the Boe-Bot - IrMultiBot.bs2
' Select one of three Boe-Bot behaviors with the IR remote 1-3 keys.

' Press POWER key to interrupt the Boe-Bot's operation.
' Then, press one of these digit keys to select a new mode:

' 1 - Control Boe-Bot with 1-9 keys and/or CH+/- and VOL+/- keys.
' 2 - Roam and avoid objects.
' 3 - Follow objects.

' Note: Startup default is mode 1.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ I/O Definitions ]----------------------------------------------

' SONY TV IR remote declaration - input receives from IR detector

IrDet          PIN     9
Speaker        PIN     4

' -----[ Constants ]---------------------------------------------------------

' SONY TV IR remote constants for non-keypad buttons

Enter          CON     11
ChUp           CON     16
ChDn           CON     17
VolUp          CON     18
VolDn          CON     19
```

```
Power          CON     21

' Boe-Bot proportional control constants (from FollowingBoeBot.bs2).

Kpl            CON     -35
Kpr            CON     35
SetPoint       CON     2
CenterPulse    CON     750

' -----[ Variables ]------------------------------------------------------

' SONY TV IR remote variables

irPulse        VAR     Word
remoteCode     VAR     Byte

' Boe-Bot navigation variables (from FollowingBoeBot.bs2).

freqSelect     VAR     Nib
irFrequency    VAR     Word
irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit
distanceLeft   VAR     Nib
distanceRight  VAR     Nib
pulseLeft      VAR     Word
pulseRight     VAR     Word

' IrMultiBot.bs2 variables.

counter        VAR     Nib                  ' <--- New
operation      VAR     Nib                  ' <--- New

' -----[ Initialization ]--------------------------------------------------

DEBUG "Press POWER to select", CR,
      "mode of operation:", CR, CR,
      "1 - Remote control 1-9/CH/VOL", CR,
      "2 - Autonomous IR roaming", CR,
      "3 - Object find and follow", CR, CR

FREQOUT Speaker, 2000, 3000

operation = 1                              ' Initialize to remote.

' -----[ Main Routine ]----------------------------------------------------

DO

  SELECT operation                         ' <--- New

    ' If operation = 1, execute a modified version of
```

```
' 7BitRemoteBoeBot.bs2 that also allows
' you to change the operation variable with the POWER key.

CASE 1                                  ' <--- New

  ' Modified main routine from 7BitRemoteBoeBot.bs2
  ' goes here.

  GOSUB Get_Ir_Remote_Code

  ' Check for POWER button.  If yes, get remote code; otherwise,
  ' send PULSOUT durations for the various maneuvers based on
  ' the value of the remoteCode variable.

  SELECT remoteCode
    CASE Power                                ' <--- New
      FREQOUT Speaker, 100, 3500              ' <--- New
      PAUSE 100                               ' <--- New
      FREQOUT Speaker, 100, 3500              ' <--- New
      PAUSE 200                               ' <--- New
      DEBUG "Select operation mode...", CR    ' <--- New
      GOSUB Get_Ir_Remote_Code                ' <--- New
      operation = remoteCode                  ' <--- New
      DEBUG ? operation, CR,                  ' <--- New
            "Running...", CR                  ' <--- New
      FREQOUT Speaker, 100, 3500              ' <--- New
      PAUSE 100                               ' <--- New
      FREQOUT Speaker, 100, 3500              ' <--- New
      PAUSE 200                               ' <--- New
    CASE 2, ChUp                            ' Forward
      PULSOUT 13, 850
      PULSOUT 12, 650
    CASE 4, VolDn                           ' Rotate Left
      PULSOUT 13, 650
      PULSOUT 12, 650
    CASE 6, VolUp                           ' Rotate Right
      PULSOUT 13, 850
      PULSOUT 12, 850
    CASE 8, ChDn                            ' Backward
      PULSOUT 13, 650
      PULSOUT 12, 850
    CASE 1                                  ' Pivot Fwd-left
      PULSOUT 13, 750
      PULSOUT 12, 650
    CASE 3                                  ' Pivot Fwd-right
      PULSOUT 13, 850
      PULSOUT 12, 750
    CASE 7                                  ' Pivot Back-left
      PULSOUT 13, 750
      PULSOUT 12, 850
    CASE 9                                  ' Pivot Back-right
```

```
        PULSOUT 13, 650
        PULSOUT 12, 750
      CASE ELSE                             ' Hold Position
        PULSOUT 13, 750
        PULSOUT 12, 750
    ENDSELECT

  ' If operation = 2, execute modified FastIrRoaming.bs2.
  CASE 2                                    ' <--- New

    ' Modified main routine from FastIrRoaming.bs2 goes here.

    IF IN9 = 0 THEN operation = 1           ' <--- New.

    FREQOUT 8, 1, 38500                     ' Check IR Detectors
    irDetectLeft = IN9
    FREQOUT 2, 1, 38500
    irDetectRight = IN0
                                                ' Decide how to navigate.
    IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
      pulseLeft = 650
      pulseRight = 850
    ELSEIF (irDetectLeft = 0) THEN
      pulseLeft = 850
      pulseRight = 850
    ELSEIF (irDetectRight = 0) THEN
      pulseLeft = 650
      pulseRight = 650
    ELSE
      pulseLeft = 850
      pulseRight = 650
    ENDIF

    GOSUB Send_Pulse                        ' <--- Modified.

  ' If operation is 3, execute the following Boe-Bot routine.
  CASE 3                                    ' <--- New

    IF IN9 = 0 THEN operation = 1           ' <--- New

    GOSUB Get_Ir_Distances

    ' Calculate proportional output.

    pulseLeft =  SetPoint - distanceLeft  * Kpl + CenterPulse
    pulseRight = SetPoint - distanceRight * Kpr + CenterPulse

    GOSUB Send_Pulse

ENDSELECT                                   ' <--- New
                                            '(End SELECT operation)
```

```
LOOP                                        ' Repeat Main Routine.
' -----[ Subroutine - Get_Ir_Remote_Code ]-------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0                            ' Clear all bits in remoteCode.

  DO                                        ' Wait for rest between messages.
    RCTIME IrDet, 1, irPulse
  LOOP UNTIL irPulse > 1000

  PULSIN IrDet, 0, irPulse                  ' Measure pulse.
  IF irPulse > 500 THEN remoteCode.BIT0 = 1  ' Set (or leave clear) bit-0.
  RCTIME IrDet, 0, irPulse                  ' Measure next pulse.
  IF irPulse > 300 THEN remoteCode.BIT1 = 1  ' Set (or leave clear) bit-1.
  RCTIME IrDet, 0, irPulse                  ' etc.
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT4 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT5 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT6 = 1

  ' Adjust remoteCode so that keypad keys correspond to the value
  ' it stores.

  IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
  IF (remoteCode = 10) THEN remoteCode = 0

  RETURN

' -----[ Subroutine - Get IR Distances ]----------------------------------

Get_Ir_Distances:
  distanceLeft = 0
  distanceRight = 0
  FOR freqSelect = 0 TO 4
    LOOKUP freqSelect,[37500,38250,39500,40500,41500], irFrequency

    FREQOUT 8,1,irFrequency
    irDetectLeft = IN9
    distanceLeft = distanceLeft + irDetectLeft

    FREQOUT 2,1,irFrequency
    irDetectRight = IN0
```

```
    distanceRight = distanceRight + irDetectRight
  NEXT
  RETURN

' -----[ Subroutine - Send_pulse ]-------------------------------------

Send_Pulse:
  PULSOUT 13,pulseLeft
  PULSOUT 12,pulseRight
  PAUSE 5
  RETURN
```

Come Funziona IrMultiBot.bs2

Come menzionato in precedenza, la routine principale seleziona una di tre diverse routine in base al valore memorizzato nella variabile **`operation`**. A ciascuna iterazione del ciclo **`DO…LOOP`**, se **`operation`** è 1, viene eseguita la routine presente nella dichiarazione **`CASE 1`**. Questa è la routine per il controllo remoto del Boe-Bot con il telecomando usando i tasti CH/VOL ed i tasti numerici. Se **`operation`** è 2, viene eseguita la routine presente nella dichiarazione **`CASE 2`**. Questa è la routine per la navigazione autonoma. Se **`operation`** è 3, viene eseguita la routine presente nella dichiarazione **`CASE 3`**. Questa è la routine per il Boe-Bot inseguitore.

```
DO
  SELECT operation

    CASE 1

      ' Modified 7BitRemoteBoeBot.bs2

      •••

    CASE 2

      ' Modified FastIrRoaming.bs2

      •••

    CASE 3

      ' Modified FollowingBoeBot.bs2

      •••

  ENDSELECT

LOOP
```

Quando il programma viene lanciato, il valore di **operation** per default viene inizializzato a zero. Questo potrebbe essere un problema perchè nessuna delle dichiarazioni **CASE** accetta uno zero. Per questa ragione, è stato aggiunto un comando per impostare ad 1 **operation** nella sezione Inizializzazione.

```
' -----[ Initialization ]------------------------------------------
   •
   •
   •
operation = 1                                  ' Initialize to remote.
```

La routine principale di 7BitRemoteBoeBot.bs2 è stata modificata per permettervi di cambiare il valore della variabile **operation** con il telecomando. Per farlo, si è dovuto aggiungere una dichiarazione **CASE** per gestire la pressione/rilascio del tasto POWER.

```
SELECT remoteCode
  CASE Power
    FREQOUT Speaker, 100, 3500
    PAUSE 100
    FREQOUT Speaker, 100, 3500
    PAUSE 200
    DEBUG "Select operation mode...", CR
    GOSUB Get_Ir_Remote_Code
    operation = remoteCode
    DEBUG ? operation, CR,
          "Running...", CR
    FREQOUT Speaker, 100, 3500
    PAUSE 100
    FREQOUT Speaker, 100, 3500
    PAUSE 200
```

Un problema è, che cosa accade se premete/rilasciate il tasto POWER quando **operation** è 2 (navigazione) o 3 (inseguitore)? In queste due modalità, non c'è modo di cambiare con il telecomando il valore della variabile **remoteCode**. Una soluzione semplice è di inserire una riga di programma all'inizion delle routine di navigazione e di inseguimento che controlli se ci sono messaggi in arrivo dal telecomando. Questa riga di codice può essere aggiunta alle altre due routine di navigazione per farle reagire alla pressione del tasto POWER.

```
IF (IN9 = 0) THEN operation = 1          ' <--- New
```

Con questa singola riga di codice, la prima cosa che le routine di navigazione e di inseguimentofanno è controllare se ci sono in arrivo messaggi dal telecomando. Se è

(IN9 = 0), allora, il valore della variabile **operation** viene impostato ad 1. Alla prossima interazione del ciclo **DO…LOOP**, verrà eseguita la routine principale modificata dal programma 7BitRemoteBoeBot.bs2. Questa routine ha una dichiarazione **CASE** per processare il tasto POWER e memorizza un nuovo valore nella variabile **operation**. Questa dichiarazione **CASE** contiene comandi che vi rendono possibile selezionare una routine differente.

Il Vostro Turno – Correggere Errori ed Aggiungere Commenti

Questo programma si perde se premete il tasto POWER, poi un tasto numerico diverso da 1, 2, o 3. Diciamo che **operation** sia impostato a 4. In questo caso, non c'è dichiarazione **CASE**, quindi il programma continua indefinitamente a ripetere il ciclo **DO…LOOP** alla ricerca di **CASE 4**, senza trovarlo. La soluzione a questo è di aggiungere una dichiarazione che acquisisca tutti i potenziali valori di **operation** e li reinvii alla routine che vi permette di selezionare con il telecomando la variabile **operation**. Un modo di farlo è con una dichiarazione **CASE ELSE**.

√ Prima di modificare questo programma, testatelo premendo il tasto POWER quindi qualsiasi tasto numerico diverso da 1, 2, o 3. Verificate che non ci sia nulla che potete fare con il telecomando per riattivare il vostro Boe-Bot. (potete premere e rilasciare il pulsante RESET sulla vostra scheda, ma con il telecomando siete sfortunati perchè non c'è nulla che potete fare per risvegliare il Boe-Bot.)

√ Localizzate gli ultimi tre comandi nella routine principale, essi dovrebbero essere simili a:

```
      GOSUB Send_Pulse

  ENDSELECT

LOOP
```

√ Aggiungete la dichiarazione **CASE ELSE** mostrata di seguito. Questo farà sì che il programma resti in ascolto per un comando dal telecomando, senza tener conto del valore di **operation**.

```
      GOSUB Send_Pulse

    CASE ELSE                                ' <--- New

      IF (IN9 = 0) THEN operation = 1          ' <--- New
```

```
    ENDSELECT

  LOOP
```


√ Testate il vostro programma medificato ed assicuratevi che il Boe-Bot non rimanga più confuso quando impostate la variabile **`operation`** ad un valore diverso da 1, 2, o 3.
√ Salvate il vostro lavoro!

ATTIVITÀ #3: BOE-BOT PROGRAMMATO A DISTANZA

per il vostro Boe-Bot potete scrivere un programma PBASIC che vi permetterà di programmarlo con schemi di movimento usando il vostro telecomando. Premendo una sequenza di tasti sul telecomando, potrete caricare una sequenza di manovre nella memoria EEPROM del BASIC Stamp del Boe-Bot. Questo è un esempio di sequenza di pressione di tasti che istruiscono il Boe-Bot ad andare in avanti per 40 impulsi, ruotare a sinistra per 20 impulsi, ruotare a destra per altri 20 impulsi, quindi andare indietro per 40 impulsi:

- POWER per inizializzare la programmazione.
- CH+, 40, ENTER per 40 impulsi in marcia avanti.
- VOL-, 20, ENTER per 20 impulsi di rotazione a sinistra.
- VOL+, 20, ENTER per 20 impulsi di rotazione a destra.
- CH-, 40, ENTER per 40 impulsi di marcia indietro.
- ENTER una seconda volta esce dalla modalità di programmazione.
- ENTER una terza volta fà eseguire le manovre al Boe-Bot.
- ENTER di nuovo fà ripetere al Boe-Bot la sequenza delle manovre.
- POWER per programmare una nuova sequenza di manovre.

Che cosa è una interfaccia utente? Si inizia dai pulsanti, It starts with the buttons, dischi combinatori, display e sistemi di menù che usate per indicare alle macchine, alle applicazioni ed ai programmi per computer che cosa volete che facciano. Probabilmente l'aspetto più importante di ogni interfaccia utente è il comportamento in risposta alla pressione del vostro pulsante, rotazione del disco combinatorio, etc.

Se l'interfaccia utente di un dato prodotto è difficile da apprendere o non è sensata, otterrà rapidamente una cattiva reputazione, e la gente non lo vorrà comprare.

L'interfaccia utente è spesso abbreviata in UI (dall'inglese User Interface).□

La parte stimolante nello scrivere codice per l'interfaccia utente UI è assicurarsi che non confonda la persona che usa il telecomando. Questo è un elenco di funzioni che il

programma dovrebbe avere per rendere il vostro Boe-Bot più semplice da programmare con un telecomando:

- Riconoscere e scartare le pressioni scorrette dei tasti.
- Uscire dalla modalità di programmazione se viene premuto due volte di seguito il tasto ENTER.
- Uscire dalla modalità di programmazione senza richiedere il numero degli impulsi.
- Permettere all'utente di ripetere molte volte la sequenza di movimenti.
- Ricordare il movimento più recente, anche se viene tolta l'alimentazione e poi ridata.

Questo è un altro progetto che dovrebbe essere suddiviso in un processo passo-passo. La maggior parte del lavoro di sviluppo verrà effettuato con l'aiuto del Terminale di Debug. Dopo che la funzionalità del programma è stata provata con il Terminale di Debug, adattarlo alla maschera per il controllo con telecomando ad infrarossi dovrebbe essere relativamente semplice.

Passo 1 – Usate il comando **`READ`** per recuperare e visualizzare i valori che erano stati memorizzati nella EEPROM in fase di compilazione con la dichiarazione **`DATA`**.

Passo 2 – Uscire da una routine senza richiedere ulteriori informazioni quando viene ricevuto un carattere di terminazione.

Passo 3 – Memorizzare valori nella EEPROM durante la fase di funzionamento con il comando **`WRITE`**, quindi recuperarli e visualizzarli.

Passo 4 – Non accettare caratteri che non hanno alcun significato per il programma; attendere fino a che il carattere giusto viene inserito.

Passo 5 – Annidare le routine di memorizzazione e recupero in un ciclo con opzioni dal menù.

Passo 6 – Adattare il prototipo del Terminale di Debug alla maschera per il controllo con telecomando ad infrarossi.

Passo 7 – Aggiungere indicatori LED ed altoparlante per aiutare l'utente.

Fase di Compilazione e Fase di Funzionamento. Il lavoro che l'editor fà con il programma prima del dowload al BASIC Stamp è fatto in fase di compilazione. Le dichiarazioni **`DATA`** e le direttive **`CON`** e **`VAR`** sono tutte processate durante la fase di compilazione. I comandi che vengono eseguiti dal BASIC Stamp mentre il programma è in funzione (**`DEBUG`**, **`FREQOUT`**, etc) lo sono durante la fase di funzionamento.

Passo 1 – Lettura e Visualizzazione di Valori Memorizzati nella EEPROM da una dichiarazione DATA

In questo passo, scriveremo e collauderemo un programma che memorizza caratteri nella EEPROM durante la fase di compilazione e li estrae ed esegue durante la fase di funzionamento. La memorizzazione verrà eseguita con la direttiva **DATA**, e l'estrazione e l'esecuzione verranno eseguite con i comandi **READ** e **DEBUG**. Ripassiamo la direttiva **DATA**; questa è la sintassi del comando presentata sulla guida sintattica del PBASIC dell'Editor del BASIC Stamp.

Sintassi: {Simbolo} DATA {@Indirizzo,} {Word} DataItem {, DataItem ...}

Per vedere le informazioni circa il comando **DATA** nella guida sintattica del PBASIC, clickate Help nell' Editor del BASIC Stamp, e selezionate Index. Digitate **DATA** nella casella di ricerca, quindi doppio click sulla voce quando appare nell'elenco in basso.

La Figura 3-4 mostra esempi delle due direttive **DATA** nel prossimo progamma esempio, DebugPlayback.bs2, e di come si relazionano alla sintassi del comando. La maggior parte di questa sintassi è stata spiegata in *Robotica con il Boe-Bot v2.0* nel Capitolo 4, Attività #6. Un elemento che può esservi nuovo è l'argomento opzionale ***{@Indirizzo}***. Viene usato nei due esempi di direttive **DATA** per memorizzare i ***DataItems*** in indirizzi specifici della EEPROM del BASIC Stamp.

Nella prima direttiva **DATA**, **@ 15** memorizza il primo ***DataItem*** all'indirizzo EEPROM 15. La "F" viene memorizzata all'indirizzo 15, la "L" all'indirizzo 16, la "R" all'indirizzo 17, etc. Il simbolo opzionale **Maneuver_List** verrà automaticamente impostato a 15 dall'Editor del BASIC Stamp durante la Fase di Compilazione. Potete usare questo simbolo nel vostro programma al posto del numero 15, e vedrete, rende molto più facile scrivere programmi per l'accesso alla EEPROM.

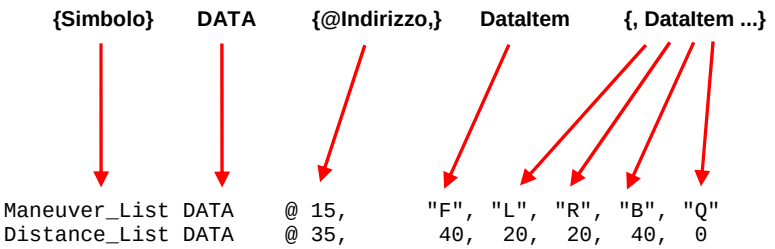


Figura 3-4 *Come può essere usata la sintassi della Direttiva* ***DATA.***

I programmi di questa attività useranno solamente valori di grandezza byte, quindi non sarà necessario il modificatore opzionale **`Word`**. Per maggiori informazioni ed esempi del modificatore **`Word`**, provate gli esempi della direttiva **`DATA`** con il modificatore **`Word`** mnella guida sintattica PBASIC dell' Editor del BASIC Stamp. Potrete anche trovare esempi che fanno uso del modificatore **`Word`** in *Che Cosa è un Microcontrollorer* ed in *Robotica con il Boe-Bot*. □

La seconda direttiva **`DATA`** usa anche l'argomento ***`@Address`***, ed i suoi ***`DataItems`*** iniziano dall' indirizzo 35. il risultato è che il numero 40 viene memorizzato all'indirizzo 35, il numero 20 all'indirizzo 36, e così via. Ambedue le direttive **`DATA`** hanno il nome simbolico opzionale***`{Simbolo}`***. Il nome simbolico per questa seconda direttiva **`DATA`** è **`Distance_List`**, e diventerà una costante 35, che userete anche in DebugPlayback.bs2.

Questa è la sintassi del comando **`READ`**:

READ *Locazione*, {*Word*} Variabile {, {*Word*} *Variabile*, ...}

`Locazione` è l'indirizzo EEPROM che memorizza il valore che volete recuperare, e ***`Variabile`*** è il nome della variabile che riceve il valore estratto dalla EEPROM. Se consultate la spiegazione di questo comando nella Guida Sintattica dell' Editor del BASIC Stamp, dice proprio questo circa ***Locazione*** e ***Variabile***:

- ***Locazione*** è una variabile/costante/espressione* (0 - 255 sul BS1, 0 - 2047 su tutti gli altri moduli BASIC Stamp) che specifica l'indirizzo della EEPROM da cui leggere.
- ***Variabile*** è una variabile (normalmente un byte) dove il valore verrà memorizzato.

La caratteristica maggiore di **Locazione** è che può essere una "variabile/costante/espressione". Una espressione è tipicamente una qualche combinazione di variabili e costanti che implicano delle operazioni matematiche.

Questo è un estratto dal prossimo programma esempio che usa le espressioni nell'argomento ***Locazione*** dei suoi comandi **READ**.

```
eeIndex = 0

DO UNTIL (direction = "Q") OR (eeIndex = 19)

  READ Maneuver_List + eeIndex, direction
  READ Distance_List + eeIndex, distance

  DEBUG direction, "  ", DEC distance, CR

  PAUSE 200
  eeIndex = eeIndex + 1

LOOP
```

Il primo comando **READ** aggiunge **Maneuver_List** (una costante uguale a 15) a **eeIndex** (una variabile il cui valore viene incrementato di uno ad ogni iterazione del ciclo **DO…LOOP**). Alla prima iterazione del ciclo **DO…LOOP**, **eeIndex** sarà zero, e **Maneuver_List** sarà sempre 15. Il risultato è che il comando READ estrae il byte presente all'indirizzo 15 della EEPROM e lo memorizza nella variabile **direction**. Dal momento che la direttiva **DATA** studiata in precedenza all'indirizzo 15 memorizzava una "F", alla prima iterazione del ciclo, il comando **READ** memorizza una "F" nella variabile **direction**.

Nella seconda iterazione, **eeIndex** sarà 1 mentre **Maneuver_List** è ancora 15. Il comando **READ** estrae la "L" memorizzato all'indirizzo 16 e lo memorizza nella variabile **direction**. Alla terza iterazione del ciclo, **eeIndex** è 2, quindi l'espressione **Location** viene calcolata in 17, e nella variabile **direction** viene memorizzato "R".

Lo stesso principio si applica al secondo comando **READ**, eccetto che **Distance_List** è 35, quindi alla prima iterazione, viene estratto il valore 40 dalla EEPROM e memorizzato in **distance**, alla seconda iterazione, 20, alla terza iterazione, ancora 20, e così via.

Programma Esempio – DebugPlayback.bs2

Questo programma esempio memorizza valori nella EEPROM del BASIC Stamp con la direttiva **DATA**, quindi con il comando **READ** recupera e visualizza questi valori.

√ Digitate e lanciate DebugPlayback.bs2.

```
' -----[ Title ]-----------------------------------------------------------
' IR Remote for the Boe-Bot - DebugPlayback.bs2
' Fetch and display DataItems from a pair of DATA directives.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ EEPROM Data ]-----------------------------------------------------

  Maneuver_List   DATA    @ 15,   "F", "L", "R", "B", "Q"
  Distance_List   DATA    @ 35,   40,  20,  20,  40,  0

' -----[ Variables ]-------------------------------------------------------

  direction    VAR     Byte
  distance     VAR     Byte
  eeIndex      VAR     Byte

' -----[ Main Routine ]----------------------------------------------------

  ' Playback routine

  eeIndex = 0

  DO UNTIL (direction = "Q") OR (eeIndex = 19)

    READ Maneuver_List + eeIndex, direction
    READ Distance_List + eeIndex, distance

    DEBUG direction, "  ", DEC distance, CR
    PAUSE 200

    eeIndex = eeIndex + 1

  LOOP

  END
```

Il Vostro Turno – Passo 2 – Uscita dalla Routine senza visualizzare Q o 0

Il ciclo **DO…LOOP** può essere modificato in modo che salti la visualizzazione della "Q" e dello 0 al termine dell'elenco. Lo potrete ottenere rimuovendo l'argomento **(direction = "Q")** dalla dichiarazione **DO UNTIL**. Quindi, aggiungete una dichiarazione **IF…THEN** con un comando **EXIT** subito dopo i comandi **READ**. Eseguendo **EXIT** quando **direction = "Q"**, il programma salterà il ciclo **DO…LOOP** prima di eseguire il comando **DEBUG**:

```
DO UNTIL (eeIndex = 19)

  READ Maneuver_List + eeIndex, direction
  READ Distance_List + eeIndex, distance

  IF (direction = "Q") THEN EXIT

  DEBUG direction, "  ", DEC distance, CR
  PAUSE 200

  eeIndex = eeIndex + 1

LOOP
```

- √ Rinominate e salvate il programma come DebugPlaybackYourTurn.bs2.
- √ Provate a modificare come mostrato il blocco di codice del ciclo **DO…LOOP** DebugPlaybackYourTurn.bs2.
- √ Lanciate il programma e verificate che non visualizzi più la "Q" e lo 0.
- √ Salvate il programma modificato.

Passo 3 – Memorizzazione di Valori durante la Fase di Funzionamento con il Comando WRITE

Quando indicate con il telecomando le direzioni del vostro Boe-Bot, memorizzerete valori nella EEPROM durante il funzionamento. Mentre la direttiva **DATA** è fatta per la memorizzazione dei dati nella EEPROM durante la fase di compilazione, il comando **WRITE** è fatto per la memorizzazione dei dati nella EEPROM durante la fase di Funzionamento. In questa attività, espanderete il programma del passo precedente in modo che possiate scrivere nella EEPROM durante la fase di funzionamento.

Mentre il comando **READ** estrae un ***DataItem*** da una ***Locazione*** nella EEPROM, il comando **WRITE** memorizza un ***DataItem*** in una ***Locazione***. Questa è la sintassi del comando **WRITE** presente nella guida Sintattica dell' Editor del BASIC Stamp:

WRITE ***Locazione, {Word} DataItem {, {Word} DataItem ...}***

Il seguente è il ciclo **DO…LOOP** che memorizza le istruzioni con i comandi **WRITE**. Aggiungendolo a DebugPlaybackYourTurn.bs2, potrete assemblare il vostro elenco di caratteri nella EEPROM durante la fase di funzionamento.

```
' Routine - Record Instructions

  eeIndex = 0

  DO UNTIL (eeIndex = 19)

    DEBUG CR, "F, B, R, L, Q", CR,
          "Enter Direction: "
    DEBUGIN direction
    WRITE Maneuver_List + eeIndex, direction

    IF direction = "Q" THEN
      DEBUG CR, CR
      EXIT
    ENDIF

    DEBUG CR, "Enter distance: "
    DEBUGIN DEC distance
    WRITE Distance_List + eeIndex, distance

    eeIndex = eeIndex + 1

  LOOP
```

Questo blocco di codice vi permetterà di impostare i valori delle variabili **direction** e **distance** con la Finestra di Trasmissione del Terminale di Debug. Ciascuno di questi valori verrà memorizzato nella EEPROM con il comando **WRITE** nello stesso modo in cui sono stati estratti dalla EEPROM in DebugPlayback.bs2.

Osserviamo attentamente i due comandi **WRITE** nel ciclo:

```
    •
    •
    •
  WRITE Maneuver_List + eeIndex, direction
    •
    •
    •
  WRITE Distance_List + eeIndex, distance
```

Dal momento che le direttive **DATA** non servirebbero, perchè nell' eseguire i comandi **WRITE** si usano ancora **Maneuver_List** e **Distance_List** nelle espressioni ***Locazione***? La risposta è che le direttive **DATA** sono ancora presenti all'inizio del programma per

riservare quelle locazioni per spazio di memoria. La sola differenza è che durante la fase di compilazione non verranno scritti ***DataItems*** nella EEPROM.

```
Maneuver_List   DATA    @ 15
Distance_List   DATA    @ 35
```

Queste due direttive **DATA** funzionano come segnaposto. **Maneuver_List** punterà ancora all'indirizzo EEPROM 15, e **Distance_List** continuerà a puntare alla locazione 35. Essi possono ancora essere usate, dai comandi **READ** e **WRITE,** come segnaposto per i due elenchi di dati EEPROM. La sola differenza è che i comandi **WRITE** memorizzeranno in queste locazioni EEPROM durante la fase di funzionamento.

Il ciclo **DO…LOOP** per la scrittura dei dati nella EEPROM ha un'altra funzione simile a quella che abbiamo aggiunto nella precedente attività della sezione Il Vostro Turno:

```
IF (direction = "Q") THEN EXIT
```

Dopo il comando **WRITE** nella **Direction_List**, la variabile direzione potrebbe ancora contenere il carattere "Q". possiamo usare questa variabile per decidere se uscire o meno dal ciclo **DO…LOOP** prima di acquisire un valore di distanza. Dal momento che in realtà dopo aver digitato "Q" non vogliamo acquisire nessuna distanza, la dichiarazione **IF…THEN** fà saltare il programma all'istruzione immediatamente seguente il comando **LOOP**.

Programma Esempio – DebugRecordPlayback.bs2

- √ Digitate e lanciate DebugRecordPlayback.bs2. (Questa è una versione modificata di DebugRecordYourTurn.bs2.)
- √ Per un risultato migliore, azionate sulla vostra tastiera il tasto Caps Lock in modo che i vostri caratteri siano tutti maiuscoli ("F", "B", "L", "R", e "Q").
- √ Seguite le richieste ed usate la Finestra di Trasmissione del Terminale di Debug peer digitare le direzioni e le distanze.
- √ Verificate che l'elenco delle direzioni e delle distanze che avete digitato sia lo stesso elenco che viene eseguito dopo aver digitato il carattere "Q".

```
' -----[ Title ]-------------------------------------------------------------
' IR Remote for the Boe-Bot - DebugRecordPlayback.bs2
' Use Debug Terminal to Store a list of values to EEPROM, then retrieve
' and display them.

' {$STAMP BS2}
' {$PBASIC 2.5}
```

```
' -----[ EEPROM Data ]-----------------------------------------------------

  Maneuver_List   DATA    @ 15
  Distance_List   DATA    @ 35

' -----[ Variables ]-------------------------------------------------------

  direction    VAR     Byte
  distance     VAR     Byte
  eeIndex      VAR     Byte

' -----[ Main Routine ]----------------------------------------------------

 ' Routine - Record Instructions

  eeIndex = 0

  DO UNTIL (eeIndex = 19)

    DEBUG CR, "F, B, R, L, Q", CR,
          "Enter Direction: "
    DEBUGIN direction
    WRITE Maneuver_List + eeIndex, direction

    IF direction = "Q" THEN EXIT

    DEBUG CR, "Enter distance: "
    DEBUGIN DEC distance
    WRITE Distance_List + eeIndex, distance

    eeIndex = eeIndex + 1

  LOOP
  DEBUG CR, CR

 ' Routine - Play Back Instructions

  eeIndex = 0
  direction = 0

  DO UNTIL (eeIndex = 19)

    READ Maneuver_List + eeIndex, direction
    READ Distance_List + eeIndex, distance

    IF direction = "Q" THEN EXIT

    DEBUG direction, "  ", DEC distance, CR
    PAUSE 200
```

```
    eeIndex = eeIndex + 1

  LOOP
  END
```


Il Vostro Turno – Passo 4 – Attendere che venga Digitato il Carattere Giusto

Un problema con l'attuale programma esempio è che accetta qualsiasi carattere, non solamente i caratteri "F", "B", "L", "R", e "Q" che a voi servono per la navigazione.

√ Lanciate nuovamente il programma e provate quando richiesto per la direzione a digitare caratteri diversi da quelli elencati.

Una dichiarazione **SELECT…CASE** all'interno di un ciclo **DO…LOOP** è uno strumento che potete usare per filtrare solamente i caratteri che volete prima di continuare. La dichiarazione **SELECT…CASE** può avere due sezioni, una con un elenco dei caratteri che volete ricevere ed un **ELSE** case per tutti gli altri caratteri che non vi interessano. Quando la variabile direzione contiene uno dei caratteri che volete, il **CASE** con l'elenco dei carattrei giusti può uscire (**EXIT)** dal ciclo **DO…LOOP**. Il blocco di codice **CASE ELSE** deve solamente contenere un messaggio che dice che è stato ricevuto il carattere sbagliato. Dopo **ENDSELECT**, il ciclo **LOOP** forzerà **DO…LOOP** a ripetersi fino a che non venga ricevuto il carattere giusto. Ecco come potete cambiare il programma in modo che filtri solamente i caratteri desiderati:

√ Rinominate e salvate il programma come DebugRecordPlaybackFiltered.bs2
√ Sostituite questi due comandi:

```
DEBUG CR, "F, B, R, L, Q", CR,
      "Enter Direction: "
DEBUGIN direction
```

Con questo blocco di codice:

```
DO
  DEBUG CR, "F, B, R, L, Q", CR,
        "Enter Direction: "
  DEBUGIN direction
  SELECT direction
    CASE "F", "B", "R", "L", "Q"
      EXIT
    CASE ELSE
      DEBUG CR, "Invalid character", CR
  ENDSELECT
LOOP
```

- √ Rilanciate il programma e verificate che accetti solamente i caratteri: "F", "B", "R", "L", e "Q".
- √ Salvate il programma modificato.

Passo 5 - Annidate le Routine di Memorizzazione e lettura in un Ciclo con le Opzioni da Menù

le routine di memorizzazione e lettura della EEPROM ora funzionano abbastanza bene. In questa attività, modificherete il programma in modo che vi dia la scelta di registrare "R" o eseguire "P" (Playback). Questi caratteri possono essere usati per scegliere tra le routine di registrazione o esecuzione. Questo è un esempio del nucleo che contiene le vostre routine di registrazione e di esecuzione:

```
DO

  DEBUG CLS,
        "R = Record", CR,
        "P = Play back", CR,
        "Choose operation: "
  DEBUGIN operation
  DEBUG CR

  IF (operation = "R") THEN

    ' Routine - Record Instructions
        •
        •
        •
  ELSEIF (operation = "P") THEN

   ' Routine - Play Back Instructions
        •
        •
        •
  ENDIF
LOOP
```

Per supportare questa routine principale dovrete dichiarare un altra variabile di grandezza byte chiamata **`operation`**. A parte questo, il prossimo programma esempio è quello che il vostro codice dovrebbe essere dopo le modifiche.

Programma Esempio – DebugRecordPlaybackWithMenu.bs2

- √ Salvate DebugRecordPlaybackFiltered.bs2 come DebugRecordPlaybackWithMenu.bs2.
- √ Modificatelo in modo che corrisponda a quello seguente.
- √ Lanciatelo.
- √ Usate il Terminale di Debug per verificare che potete digitare "R" per registrare direzioni e distanze e "P" per eseguirle.
- √ Verificate anche che qualsiasi altro carattere farà visualizzare il messaggio "Invalid character, try again".
- √ Salvate il vostro programma.

```
' -----[ Title ]-----------------------------------------------------------
' IR Remote for the Boe-Bot - DebugRecordPlaybackWithMenu.bs2
' Use the Debug Terminal to select between record and playback modes.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ EEPROM Data ]-----------------------------------------------------

  Maneuver_List   DATA    @ 15
  Distance_List   DATA    @ 35

' -----[ Variables ]-------------------------------------------------------

  direction    VAR     Byte
  distance     VAR     Byte
  eeIndex      VAR     Byte
  operation    VAR     Byte

' -----[ Main Routine ]----------------------------------------------------

DO

  DEBUG CLS,
        "R = Record", CR,
        "P = Play back", CR,
        "Choose operation: "
  DEBUGIN operation
  DEBUG CR

  IF (operation = "R") THEN

    ' Routine - Record Instructions

    eeIndex = 0
    direction = 0
```

```
   DO UNTIL (eeIndex = 19)

     DO

       DEBUG CR, "F, B, R, L, Q", CR,
                 "Enter Direction: "
       DEBUGIN direction

       SELECT direction
         CASE "F", "B", "R", "L", "Q"
           EXIT
         CASE ELSE
           DEBUG CR, "Invalid character", CR
       ENDSELECT

     LOOP

     WRITE Maneuver_List + eeIndex, direction

     IF (direction = "Q") THEN EXIT

     DEBUG CR, "Enter distance: "
     DEBUGIN DEC distance
     WRITE Distance_List + eeIndex, distance

     eeIndex = eeIndex + 1

   LOOP

   DEBUG CR

 ELSEIF (operation = "P") THEN

' Routine - Play Back Instructions

 eeIndex = 0
 direction = 0

 DO UNTIL (eeIndex = 19)

   READ Maneuver_List + eeIndex, direction
   READ Distance_List + eeIndex, distance

   IF direction = "Q" THEN EXIT

   DEBUG direction, "  ", DEC distance, CR
   PAUSE 200

   eeIndex = eeIndex + 1
```

```
  LOOP

  ELSE

    DEBUG "Invalid character", CR,
          "try again."

  ENDIF

  DEBUG CR, "Press any key..."
  DEBUGIN operation

LOOP
```

Il Vostro Turno – Passo 6 – Adattare le manovre al Boe-Bot

Potete sostituire il comando **`PAUSE 200`** con una semplice routine per pilotare il Boe-Bot. Ecco come:

- √ Rinominate e salvate con il nome di DebugRecordBoeBotPlayback.bs2.
- √ Aggiungete questa dichiarazione di variabile nella Sevione Variabili del programma:

```
pulseCount   VAR     Byte
```

- √ Sostituire il comando **`PAUSE 200`** con questo blocco di codice:

```
FOR pulseCount = 1 TO distance

  SELECT direction
    CASE "F"
      PULSOUT 13, 850
      PULSOUT 12, 650
    CASE "B"
      PULSOUT 13, 650
      PULSOUT 12, 850
    CASE "L"
      PULSOUT 13, 650
      PULSOUT 12, 650
    CASE "R"
      PULSOUT 13, 850
      PULSOUT 12, 850
  ENDSELECT

  PAUSE 20

NEXT
```

- √ Lanciate il programma e verificate che possiate programmare e riprogrammare la sequenza dei movimenti con il Terminale di Debug.
- √ Salvate il vostro programma modificato.

Passo 7 – Adattamento del programma al Telecomando ad Infrarossi

Aquesto punto, siete abbastanza vicini al completamento della realizzazione di un Boe-Bot programmabile con telecomando. Potete usare la routine principale del Passo 6, di DebugRecordBoeBotPlayback.bs2, e, dopo opportuni adattamenti, inserirlo nel programma IrRemoteKeypad.bs2. questa è un elenco che vi fornisce una idea generale degli adattamenti che debbono essere fatti:

- Al posto di **`DEBUGIN`** si usa **`GOSUB Get_Ir_Remote_Code`**.
- Al posto di **`DEBUGIN DEC`** si usa **`GOSUB Get_Multi_Digit_Value`**.
- **`ChUp`**, **`ChDn`**, **`VolUp`**, e **`VolDn`** vengono usati invece di "F", "B", "R", ed "L".
- Si usa **`ENTER`** al posto di "Q".
- **`POWER`** ed **`ENTER`** vengono usati invece di "P" ed "R".

Segue una descrizione dettagliata di come DebugRecordBoeBotPlayback.bs2 viene adattato ed integrato in IrRemoteKeypad.bs2:

- √ Aprite IrRemoteKeypad.bs2 e salvatene una copia con il nome di RemoteRecordBoeBotPlayback.bs2.
- √ Aggiornate la Sezione Titolo per includere il nome corretto del programma, una breve descrizione, e le istruzioni di uso:

```
' -----[ Title ]-------------------------------------------------------
' IR Remote for the Boe-Bot - RemoteRecordBoeBotPlayback.bs2
' Press key sequences to program motion routines into the
' Boe-Bot's EEPROM and replay them.

' Press POWER key to program or ENTER key to play program.
' In programming mode, press a CH/VOL key to choose a maneuver.
' Use the keypad to enter the number of pulses, then press ENTER.
' Pressing ENTER again terminates programming.
' Pressing ENTER a third, fourth, etc time replays the program.
' Press POWER to reprogram.

' {$STAMP BS2}
' {$PBASIC 2.5}
```

√ Inserite questa sezione tra la sezione direttive Stamp/PBASIC e la sezione Definizioni I/O.

```
' -----[ EEPROM Data ]-----------------------------------------------

' Set aside data for lists of Maneuver_List and Distance_List.

Maneuver_List DATA @ 15                 ' 20 bytes for Maneuver_List.
Distance_List DATA @ 35                 ' 20 bytes for Distance_List.
```

√ Copiate queste dichiarazioni nella sezione Definizioni dei Piedini:

```
' Boe-Bot Servo Pins

ServoLeft      PIN     13
ServoRight     PIN     12
```

√ Copiate queste dichiarazioni nella sezione Variabili:

```
' Boe-Bot navigation variables

direction      VAR     Byte
counter        VAR     Word
eeIndex        VAR     Byte
```

Se esaminate attentamente il prossimo ciclo **DO…LOOP**, vedrete che ha la stessa funzione generale e la stessa struttura della routine principale di DebugRecordBoeBotPlayback.bs2. □

√ Sostituire il ciclo **DO…LOOP** della routine principale con questo.

```
DO

  DEBUG "Press POWER to record", CR,
        "or ENTER to playback", CR, CR

  GOSUB Get_Ir_Remote_Code

  IF remoteCode = Power THEN

    ' Routine - Record data

    GOSUB Beep_Valid

    eeIndex = 0

    DO UNTIL eeIndex = 19

      DEBUG "Menu: ", CR,
```

```
            "  CH+   = Forward", CR,
            "  CH-   = Backward", CR,
            "  VOL+  = Right", CR,
            "  VOL-  = Left", CR,
            "  ENTER = Done recording", CR, CR

      DO

        GOSUB Get_Ir_Remote_Code

        SELECT remoteCode
          CASE ChUp TO VolDn, Enter
            GOSUB Beep_Valid
            EXIT
          CASE ELSE
            DEBUG "Press CH+/-, VOL+/-, or ENTER", CR
            GOSUB Beep_Error
        ENDSELECT

      LOOP

      direction = remoteCode
      WRITE eeIndex + Maneuver_List, direction

      IF (direction = Enter) THEN EXIT

      DEBUG "Enter number of pulses: ", CR
      GOSUB Get_Multi_Digit_VAlue
      DEBUG DEC value, " pulses", CR, CR
      WRITE eeIndex + Distance_List, value
      eeIndex = eeIndex + 1

    LOOP

  ELSEIF remoteCode = Enter THEN

    ' Routine - playback data

    GOSUB Beep_Valid
    DEBUG "Running...", CR, CR

    eeIndex = 0
    direction = 0

    DO UNTIL (eeIndex = 19)

      READ eeIndex + Maneuver_List, direction
      READ eeindex + Distance_List, value

      IF (direction = Enter) THEN EXIT
```

```
      FOR counter = 1 TO value
        SELECT direction
          CASE ChUp
            PULSOUT ServoLeft, 850
            PULSOUT ServoRight, 650
          CASE ChDn
            PULSOUT ServoLeft, 650
            PULSOUT ServoRight, 850
          CASE VolUp
            PULSOUT ServoLeft, 850
            PULSOUT ServoRight, 850
          CASE VolDn
            PULSOUT ServoLeft, 650
            PULSOUT ServoRight, 650
        ENDSELECT
        PAUSE 20
      NEXT

      eeIndex = eeIndex + 1

    LOOP

  ELSE

    GOSUB Beep_Error

  ENDIF

LOOP
```

Programma Esempio – RemoteRecordBoeBotPlayback.bs2

Viene di seguito mostrato il programma completato.

- √ Digitate e lanciate il programma e seguite le richieste del Terminale di Debug per la programmazione remota.
- √ Provate la seguente sequenza di tasti:
  - √ POWER per iniziare la programmazione.
  - √ CH+, 40, ENTER per andare a marcia avanti per 40 impulsi.
  - √ VOL-, 20, ENTER per ruotare a sinistra per 20 impulsi.
  - √ VOL+, 20, ENTER per ruotare a destra per 20 impulsi.
  - √ CH-, 40, ENTER per andare a marcia indietro per 40 impulsi.
  - √ ENTER una seconda volta per uscire dalla modalità programmazione.
  - √ ENTER una terza volta per far eseguire le manovre al Boe-Bot.
  - √ ENTER di nuovo fà ripetere al Boe-Bot la sequenza delle manovre.

√ POWER per programmare una nuova sequenza di manovre.

```
' -----[ Title ]-----------------------------------------------------
' IR Remote for the Boe-Bot - RemoteRecordBoeBotPlayback.bs2
' Press key sequences to program motion routines into the
' Boe-Bot's EEPROM and replay them.

' Press POWER key to program or ENTER key to play program.
' In programming mode, press a CH/VOL key to choose a maneuver.
' Use the keypad to enter the number of pulses, then press ENTER.
' Pressing ENTER again terminates programming.
' Pressing ENTER a third, fourth, etc time replays the program.
' Press POWER to reprogram.

' {$STAMP BS2}                          ' $STAMP directive
' {$PBASIC 2.5}                         ' $PBASIC directive

' -----[ EEPROM Data ]-----------------------------------------------

' Set aside data for lists of Maneuver_List and Distance_List.

Maneuver_List DATA @ 15                 ' 20 bytes for Maneuver_List.
Distance_List DATA @ 35                 ' 20 bytes for Distance_List.

' -----[ I/O Definitions ]-------------------------------------------------

' SONY TV IR remote declaration - input receives from IR detector

IrDet          PIN     9
Speaker        PIN     4

' Boe-Bot Servo Pins

ServoLeft      PIN     13
ServoRight     PIN     12

' -----[ Constants ]-------------------------------------------------------

' SONY TV IR remote constants for non-keypad buttons.

Enter          CON     11
ChUp           CON     16
ChDn           CON     17
VolUp          CON     18
VolDn          CON     19
Power          CON     21

' -----[ Variables ]-------------------------------------------------------

' SONY TV IR remote variables
```

```
irPulse        VAR     Word                ' Single-digit remote variables
remoteCode     VAR     Byte
index          VAR     Nib
value          VAR     Word                ' Stores multi-digit value

' Boe-Bot navigation variables

direction      VAR     Byte
counter        VAR     Word
eeIndex        VAR     Byte

' -----[ Main Routine ]------------------------------------------------

DO

  DEBUG "Press POWER to record", CR,
        "or ENTER to playback", CR, CR

  GOSUB Get_Ir_Remote_Code

  IF remoteCode = Power THEN

    ' Routine - Record data

    GOSUB Beep_Valid

    eeIndex = 0

    DO UNTIL eeIndex = 19

      DEBUG "Menu: ", CR,
            "  CH+   = Forward", CR,
            "  CH-   = Backward", CR,
            "  VOL+  = Right", CR,
            "  VOL-  = Left", CR,
            "  ENTER = Done recording", CR, CR

      DO

        GOSUB Get_Ir_Remote_Code

        SELECT remoteCode
          CASE ChUp TO VolDn, Enter
            GOSUB Beep_Valid
            EXIT
          CASE ELSE
            DEBUG "Press CH+/-, VOL+/-, or ENTER", CR
            GOSUB Beep_Error
        ENDSELECT
```

```
    LOOP

    direction = remoteCode
    WRITE eeIndex + Maneuver_List, direction

    IF (direction = Enter) THEN EXIT

    DEBUG "Enter number of pulses: ", CR
    GOSUB Get_Multi_Digit_Value
    DEBUG DEC value, " pulses", CR, CR
    WRITE eeIndex + Distance_List, value
    eeIndex = eeIndex + 1

  LOOP

ELSEIF remoteCode = Enter THEN

  ' Routine - playback data

  GOSUB Beep_Valid
  DEBUG "Running...", CR, CR

  eeIndex = 0
  direction = 0

  DO UNTIL (eeIndex = 19)

    READ eeIndex + Maneuver_List, direction
    READ eeindex + Distance_List, value

    IF (direction = Enter) THEN EXIT

    FOR counter = 1 TO value
      SELECT direction
        CASE ChUp
          PULSOUT ServoLeft, 850
          PULSOUT ServoRight, 650
        CASE ChDn
          PULSOUT ServoLeft, 650
          PULSOUT ServoRight, 850
        CASE VolUp
          PULSOUT ServoLeft, 850
          PULSOUT ServoRight, 850
        CASE VolDn
          PULSOUT ServoLeft, 650
          PULSOUT ServoRight, 650
      ENDSELECT
      PAUSE 20
    NEXT

    eeIndex = eeIndex + 1
```

```
    LOOP

  ELSE

    GOSUB Beep_Error

  ENDIF

LOOP

' -----[ Subroutine - Get_Ir_Remote_Code ]--------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0                            ' Clear all bits in remoteCode.

  DO                                        ' Wait for rest between messages.
    RCTIME IrDet, 1, irPulse
  LOOP UNTIL irPulse > 1000

  PULSIN IrDet, 0, irPulse                  ' Measure pulse.
  IF irPulse > 500 THEN remoteCode.BIT0 = 1  ' Set (or leave clear) bit-0.
  RCTIME IrDet, 0, irPulse                  ' Measure next pulse.
  IF irPulse > 300 THEN remoteCode.BIT1 = 1  ' Set (or leave clear) bit-1.
  RCTIME IrDet, 0, irPulse                  ' etc.
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT4 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT5 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT6 = 1

  ' Adjust remoteCode so that keypad keys correspond to the value
  ' it stores.

  IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
  IF (remoteCode = 10) THEN remoteCode = 0

  RETURN

' -----[ Subroutine - Get_Multi_Digit_Value ]-----------------------------

' Acquire multi-digit value (up to 65535) and store it in
' the value variable.  Speaker beeps each time a key is
```

```
' pressed.

Get_Multi_Digit_Value:

  value = 0
  remoteCode = 0

  DO

    value = value * 10 + remoteCode

    DO
      GOSUB Get_Ir_Remote_Code
      IF (remoteCode < 10) THEN
        DEBUG "You pressed: ", DEC1 remoteCode, CR
        GOSUB Beep_Valid
        EXIT
      ELSEIF (remoteCode = Enter) THEN
        DEBUG "You pressed: ENTER", CR
        GOSUB Beep_Valid
        EXIT
      ELSE
        DEBUG "Press 0-9 or ENTER", CR
        GOSUB Beep_Error
      ENDIF
    LOOP

  LOOP UNTIL (remoteCode = Enter)

  RETURN

' -----[ Subroutine - Beep_Valid ]-----------------------------------

' Call this subroutine to acknowledge a key press.

Beep_Valid:

  FREQOUT Speaker, 100, 3500
  PAUSE 200

  RETURN

' -----[ Subroutine - Beep_Error ]-----------------------------------

' Call this subroutine to reject a key press.

Beep_Error:

  FREQOUT Speaker, 100, 3000
  PAUSE 200
  RETURN
```

Il Vostro Turno – Avvisi Ottici con LED ed Organizzazione del Programma

Per rendere completo questo programma, un paio di LED possono facilitare la programmazione del Boe-Bot senza il Terminale di Debug. Questo può essere utile in alcuni contesti di navigazione.

- √ Rinominate il vostro programma come RemoteRecordBoeBotPlaybackLed.bs2.
- √ Progettate ed implementate gli avvisi ottici con LED che coincidano con le informazioni del T>erminale di Debug.
- √ Scrivete un manuale di istruzioni per il vostro Boe-Bot descrivendo come usare il telecomando per programmare il Boe-Bot affidandosi solamente sull'altoparlante e sui LED. Usate il foglio istruzioni che accompagnava il telecomando universale come esempio.

Le routine di programmazione e di funzionamento dovrebbero essere ttrasformate in subroutine.

- √ Fatelo, quindi salvate il vostro lavoro. Potreste volerle usare in altri programmi.

SOMMARIO

In questo capitolo sono stati studiate altre applicazioni che potete realizzare con l'applicazione per il telecomando IR. I programmi applicativi devono essere ben organizzati, con ciascun compito eseguito in una subroutine. Anche i piedini I/O, le variabili e le costanti devono essere definite e documentate nella Sezione Dichiarazioni. Quando i programmi applicativi seguono queste convenzioni, è possibile combinare i due (o più) di questi programmi applicativi per eseguire obbiettivi robotici più avanzati e complessi.

Le prime due attività sono state incentrate su come mescolare ed integrare più di un programma applicazitivo in un programma più complesso che esegue più compiti. In ciascuna di queste attività, le subroutine dei vari programmi vengono copiate ed incollate in programmi più ampi, si opera in modo simile con le costanti, le variabili ed altre dichiarazioni. In base al fatto che i programmi applicativi sono scritti in forma modulare, non si è dovuto modificare nessuna delle subroutine. Invece, esse sono utilizzate ed orchestrate nella routine principale. Affidandosi alla funzionalità delle subroutine, la routine principale può essere scritta in termini più semplici e che eseguono compiti più complessi.

Nelle prime due attività, è stata enfatizzato il collaudo dei programmi e dei circuiti che sono stati usati in *Robotica con il Boe-Bot*. Si deve esssere sicuri che ciascun programma e circuito funzionino individualmente prima che possano essere combinati in un circuito e/o programma più grande. È stato presentato **`SELECT…CASE`** come mezzo di scelta tra opzioni alternative di routine principale con un sistema di menù. Questo schema è facile da espandere semplicemente aggiungendo altre dichiarazioni **`CASE`**.

Le subroutine di un singolo schema di applicazione possono anche essere usate in modi nuovi e creativi per creare ulteriori comportamenti robotici e più potenti. La terza attività lo ha dimostrato costruendo uno schema di inserimento da tastiera di telecomando IR per la programmazione del Boe-Bot. È stata esaminata più attentamente la memoria EEPROM come utensile per la memorizzazione ed il recupero di sequenze di caratteri e valori. Usando l'operatore **`@Address`** in una direttiva **`DATA`**, potete definire blocchi di memoria non usata come memoria di servizio del programma. Nomi simbolici opzionali ***`Simbolo`*** possono essere assegnati ai comandi **`DATA`** per aiutare il calcolo della locazione di un dato **`DataItem`**. È resa molto più semplice l'uso della **`EEPROM`** con il comando **`WRITE`** per la memorizzazione ed il comando **`READ`** per la lettura. Specialmente se dovete

memorizzare e gestire più di un elenco di elementi correlati, come ad esempio le manovre e le distanze del Boe-Bot.

In questo testo sono stati studiati gli elementi di progettazione delle interfaccie utenti (User Interface UI). Dai capitoli precedenti, sono stati presentati ed applicati, la funzione di antirimbalzo dei pulsanti e la retroazione con altoparlante e LED. Nella terza attività di questo capitolo, sono state studiate le tecniche di programmazione per limitare il numero delle pressioni di tasti che il programma accetta. Tramite esempi sono stati studiati la progettazione e l'implementazione di sequenze sensate di pressioni di tasti per la scelta di specifiche funzioni robotiche.

È anche stato studiato un rocesso passo-passo per la soluzione di funzioni complesse. Nella progettazione della interfaccia utente, si è fatto grande uso delle finestre di Trasmissione e di Ricezione del Terminale di Debug come strumento di prototipazione. Dopo aver definito le interazioni con l'utente con l'aiuto del Terminale di Debug, sono stati importati in una maschera di comunicazione remota ad IR.

Domande

1. Qual'è il nome della variabile usata per il controllo della velocità del programma IrRoamingWithSpeedControl.bs2?
2. Che cosa fà **`CLS`** ne comando **`DEBUG CLS`**?
3. Che cosa accade quando la variabile usata in una dichiarazione **`SELECT…CASE`** non contiene nessuno dei valori specificati nella dichiarazione **`CASE`**? Se questo diventa un problema come lo potreste risolvere?
4. Che cosa fà l'argomento opzionale ***`@Address`*** in una direttiva **`DATA`**? In che modo questo influisce sul valore del ***`Simbolo`*** opzionale?
5. qual'è la differenza tra il comando **`READ`** ed il comando **`WRITE`**?
6. Come può essere usata una dichiarazione **`SELECT…CASE`** all'interno di un ciclo **`DO…LOOP`** per attendere un valore specifico prima di continuare con il passo successivo del programma?

Esercizi

1. In IrRoamingWithSpeedControl.bs2, quali saranno le durate degli impulsi per i servo sinistro e destro se nessun oggetto viene rilevato e la velocità = 50?

2. La scelta nella sezione routine principali nell' attività #2 mostra il nucleo della dichiarazione **`SELECT…CASE`**. Espandete questo nucleo in maniera che contenga un caso per 4 valori ed un caso alternativo per qualsiasi altro valore.
3. Scrivete comandi **`READ`** per recuperare i valori 7 e 13 usando **`My_List`** nell'argomento ***`Locazione`***. Assumete di aver dichiarato una variabile = **`myValue`**.
4. Modificate questo blocco di codice in modo che accetti solamente valori tra 16 e 19.

```
DO
  DEBUG CR, "F, B, R, L, Q", CR,
        "Enter Direction: "
  DEBUGIN direction
  SELECT direction
    CASE "F", "B", "R", "L", "Q"
      EXIT
    CASE ELSE
      DEBUG CR, "Invalid character", CR
  ENDSELECT
LOOP
```

5. Con RemoteRecordBoeBotPlayback.bs2 elencate la sequenza di pressioni dei tasti del telecomando per far percorrere al Boe-Bot un rettangolo di 50 impulsi per 100.

Progetti

1. Usate 7BitRemoteBoeBot.bs2 dal capitolo 2, Attività #3 come punto di partenza. Aggiungete una routine che annulli le istruzioni del telecomando e fermi il Boe-Bot prima che collida con un oggetto. In questo modo, potrete provare a guidare il Boe-Bot contro un ostacolo, ma il porgramma non ve lo permetterà.
2. Espandete IrMultiBotYourTurn.bs2 in modo che potete premere i seguenti tasti numerici per le seguenti funzioni:
    - 1 - Boe-Bot controllato da remoto
    - 2 - Navigazione IR alla masssima velocità
    - 3 - Boe-Bot Inseguitore
    - 4 - Modalità di centratura dei servo
    - 5 - Navigazione a bassa velocità per il Boe-Bot guida
    - 6 – Rivelatore di interferenza a lampeggio singolo

Soluzioni

Q1. Il nome della variabile è **speed**.

Q2. Secondo la Guida Sintattica del PBASIC, esso pulisce lo schermo. In altre parole, esso cancella qualsiasi cosa attualmente visualizzata nella finestra di Trasmissione del Terminale di Debug.

Q3. Se nessuna delle dichiarazioni **CASE** corrisponde alla variabile di **SELECT** il programma cerca la parola chiave **ENDSELECT** e continua da lì. Potete aggiungere una dichiarazione alternativa **CASE ELSE** con qualsiasi codice che volete che sia eseguito dalla dichiarazione **SELECT…CASE** se la variabile che è stata selezionata contiene un valore che non corrisponde a nessuna delle altre dichiarazioni **CASE**.

Q4. L'operatore ***@Address*** vi consente di specificare l'indirizzo di partenza della EEPROM per il primo ***DataItem*** della direttiva **DATA**. Il ***Simbolo*** per quella direttiva data sarà una costante uguale al valore che segue l'operatore ***@Address***. Questo sarà anche l'indirizzo del primo ***DataItem*** nella direttiva **DATA**.

Q5. Il comando **WRITE** contiene un ***DataItem*** in una ***Locazione*** nella EEPROM; il comando **READ** estrae un ***DataItem*** da una ***Locazione*** della EEPROM.

Q6. Un ciclo **DO…LOOP** può avere una dichiarazione **SELECT…CASE** annidata al suo interno. Una delle dichiarazioni **CASE** può contenere un elenco dei possibili valori che volete sostituire prima di continuare con il programma. Il blocco di codice per questa dichiarazione **CASE** dovrebbe contenere il comando **EXIT** per interrompere il ciclo **DO…LOOP** dal momento che il suo blocco di codice verrà eseguito quando uno dei valori desiderati è contenuto dalla variabile della dichiarazione **SELECT**. L'altra dichiarazione **CASE ELSE** dovrebbe contenere un blocco di codice che verrà eseguito quando si riceverà il valore sbagliato. Dal momento che non contiene un comando **EXIT**, il ciclo **DO…LOOP** si ripeterà fino a che uno dei valori desiderati non verrà memorizzato nella variabile della dichiarazione **SELECT**.

E1. Se non viene rilevato alcun oggetto, verrà eseguita la dichiarazione **CASE ELSE**. Questa case imposta le variabili usate per i comandi **PULSOUT** dei servo sinistro e destro. Il valore di **pulseLeft** sarà 750 – 50 = 700. Per calcolare la durata dell'impulso, moltiplicate l'argomento durata del comando **PULSOUT** per 2 µs. 700 × 0.000002 s = 0.0014 s = 1.4 ms. Il valore di **pulseRight** sarà 750 + 50 = 800, che invia un impulso di 1.6 ms.

E2. Accanto al ciclo **DO…LOOP** della routine principale, c'è un **ENDSELECT**. Inserite questo blocco di codice subito prima di **ENDSELECT**.

```
' If operation is 4, execute the YourProgram.bs2.
CASE 4

   ' Modified main routine from YourProgram.bs2 goes here.
        •
        •
        •
' If operation is not 1-4, execute this routine.
CASE ELSE

   ' Catch-all commands go there.
        •
        •
        •
```

E3. Soluzione: **READ My_List + 4, myValue: READ My_List + 7, myValue**

E4. Ecco il codice modificato. Fate attenzione nell'aggiungere l'operatore **DEC** al comando **DEBUGIN**.

```
DO
  DEBUG CR, "16, 17, 18, or 19", CR,
        "Enter Value: "
  DEBUGIN DEC direction
  SELECT direction
    CASE 16 to 19
      EXIT
    CASE ELSE
      DEBUG CR, "Invalid character", CR
  ENDSELECT
LOOP
```

E5. Provate questa sequenza con alcuni aggiustamenti per ottenere nella curva di 21 impulsi, con i vostri servo, una rotazione di 90 gradi:

- √ POWER per iniziare la programmazione.
- √ CH+, 100, ENTER per la marcia in avanti con 100 impulsi.
- √ VOL-, 21, ENTER per la curva a sinistra con 21 impulsi.
- √ CH+, 50, ENTER per la marcia in avanti con 50 impulsi.
- √ VOL-, 21, ENTER per la curva a sinistra con 21 impulsi.
- √ CH+, 100, ENTER per la marcia in avanti con 100 impulsi.
- √ VOL-, 21, ENTER per la curva a sinistra con 21 impulsi.
- √ CH+, 50, ENTER per la marcia in avanti con 50 impulsi.

√ ENTER una seconda volta per uscire dalla modalità programmazione.
√ ENTER una terza volta per far eseguire le manovre al Boe-Bot.

Soluzioni dei Progetti

P1. Aaggiungete queste dichiarazioni bit alla sezione delle Variabili del programma.

```
irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit
```

√ Selezionate e copiate le funzioni di rilevazione IR dal programma TestIrPairsAndIndicators.bs2 ed incollatele nel vostro progetto di programma subito prima della dichiarazione **SELECT** nella routine principale.

```
FREQOUT 8, 1, 38500
irDetectLeft = IN9

FREQOUT 2, 1, 38500
irDetectRight = IN0
```

√ Modificate questa dichiarazione **CASE**:

```
CASE 2, ChUp
  PULSOUT 13, 850
  PULSOUT 12, 650
```

In modo che sia

```
CASE 2, ChUp
  IF (irDetectLeft = 1) AND (irDetectRight = 1) THEN
    PULSOUT 13, 850
    PULSOUT 12, 650
  ELSE
    FREQOUT Speaker, 3, 4500
  ENDIF
```

P2. Aggiungete dichiarazioni case per ciascuna funzione nella **SELECT…CASE** esterna del ciclo **DO…LOOP** della routine principale. Ricordate di aggiornare l'elenco del menù nella routine di Inizializzazione.

```
DO
  SELECT operation

    CASE 1
          •
          •
```

```
      •
CASE 2
      •
      •
      •
CASE 3
      •
      •
      •
' If operation is 4, put Boe-Bot in servo centering mode.
CASE 4

  IF IN9 = 0 THEN operation = 1

  pulseLeft = 750
  pulseRight = 750

  GOSUB Send_Pulse

' If operation is 5, run slow roaming routine for the lead Boe-Bot
CASE 5

  ' Modified main routine from FastIrRoaming.bs2 goes here.

  IF IN9 = 0 THEN operation = 1

  FREQOUT 8, 1, 38500                    ' Check IR Detectors
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  irDetectRight = IN0
                                         ' Decide how to navigate.
  IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
    TOGGLE 1
    TOGGLE 10
    pulseLeft = 725
    pulseRight = 775
  ELSEIF (irDetectLeft = 0) THEN
    TOGGLE 10
    pulseLeft = 775
    pulseRight = 775
  ELSEIF (irDetectRight = 0) THEN
    TOGGLE 1
    pulseLeft = 725
    pulseRight = 725
  ELSE
    pulseLeft = 775
    pulseRight = 725
  ENDIF

  GOSUB Send_Pulse                       ' <--- Modified.
```

```
      LOW 1
      LOW 10

    ' If operation is 6, test for ballast interference.  This versions
    ' of the ballast interference tester only triggers once.  If
    ' triggered, the remote has to be used to reset the Boe-Bot to
    ' function 6 to repeat the 1-shot ballast interference detection.
     CASE 6

      DO

        IF IN9 = 0 THEN

        operation = 1

        FOR counter = 1 TO 10
          FREQOUT 4, 40, 4500
          PAUSE 50
        NEXT

        EXIT

        ENDIF

      LOOP

    CASE ELSE

      IF IN9 = 0 THEN operation = 1          ' <--- New

  ENDSELECT                                  ' <--- New
                                             '(End SELECT operation)
LOOP                                         ' Repeat Main Routine.
```

Appendice A: Documentazione del Kit Applicativo per Telecomando IR

PARALLAX INC

599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com
Educational: www.stampsinclass.com

INFRARED REMOTE APPKIT (#29112)

Una Tastiera senza fili per il Vostro Modulo Microcontrollore BASIC Stamp®

Con un telecomando universale ed un ricevitore infrarosso, potrete aggiungere una tastiera senza fili alle vostre applicazioni con il BASIC Stamp. Il ricevitore IR è poco costoso, ed occupa solamente un piedino I/O. Anche i telecomandi universali costano poco, facili da trovare e da sostituire, inoltre hanno abbastanza pulsanti per la maggior parte delle applicazioni. I componenti di questo kit insieme ai programmi esempio rendono possibile l'inserimento di valori e controllare i vostri progetti esattamente come fareste con un TV, VCR, od altri componenti del sistema di intrattenimento.

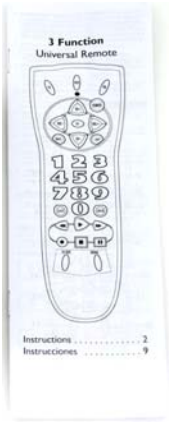
Il telecomando IR Remoto può anche aggiungere una marcia in più ai vosti progetti robotici. Mentre l'inserto di questa confezione vi fornisce le informazioni di base essenziali, i circuiti, ed i programmi esempio per iniziare, potete apprendere molto di più con *Inffrarossi per il Controllo a Distanza con ilBoe-Bot*. Questo testo è principalmente la continuazione di *Robotica con il Boe-Bot*, ma con il vantaggio del controllo remoto. Segue lo stesso formato in termini di presentazione di nuovo hardware, spiegare come funzionano le cose, e dimostrando nuove tecniche PBASIC. Le applicazioni IR remote per il robot Boe-Bot™ includono il controllo remoto, il controllo tramite tastiera, il comportamento ibrido autonomo e controllo remoto e la programmazione di sequenze da telecomando.

Contenuto del Kit *

Elenco Componenti:

(1) 020-00001 Telecomando Universale con Manuale
(1) 350-00014 Rivelatore IR
(1) 150-02210 Resistenza – 220 Ω
(1) 800-00016 Ponticelli in filo rigido– busta di 10

*Due pile alkaline AA vendute separatamente

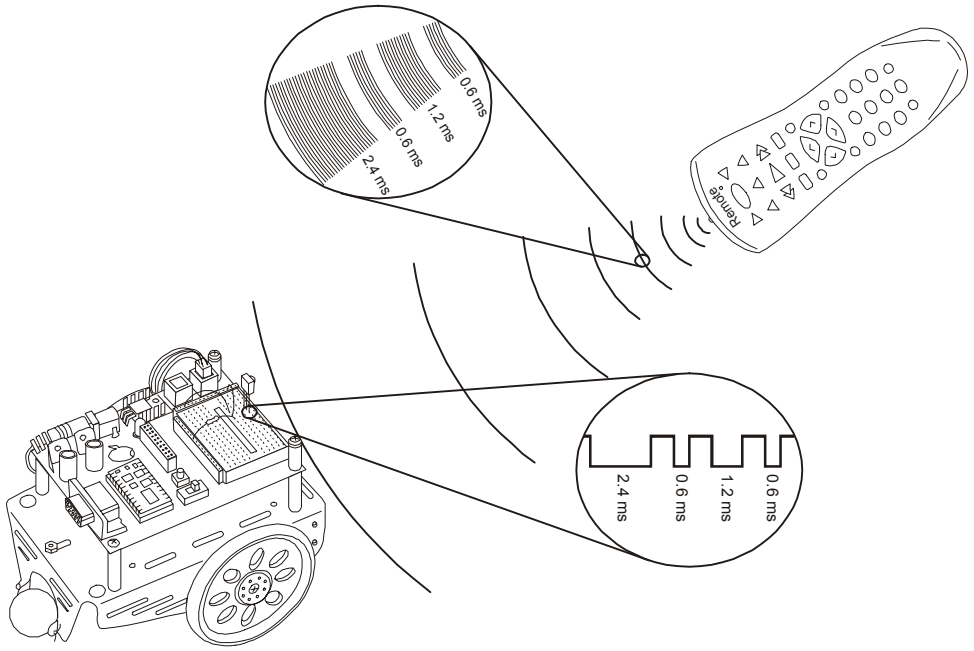


Come Funziona la Comunicazione IR

Il telecomando universale invia messaggi emettendo brevi lampeggi di 38.5 kHz dai LED IR. Il dato effettivo è contenuto nella durata di ciascun impulso. Esistono molti protocolli IR diversi, ma, in generale la durata del segnale a 38.5 kHz trasmette un qualche genere di messaggio. Una durata può indicare l'inizio del messaggio, mentre un altro indica un 1 binario, ed ancora un altro indica uno 0 binario.

Il piedino di uscita del rivelatore IR invia un segnale basso se rileva la presenza del segnale IR a 38.5 kHz, ed un segnale alto se non lo rileva. In questo modo, un segnale basso di una certa durata può indicare l'inizio del messaggio, mentre un altro indica un 1 binario, ed un altro ancora indica uno 0 binario. Questo tipo di comunicazione viene chiamato modulazione a larghezza di impulso (PWM dall'inglese pulse width

modulation), perchè quando graficizzato con il tempo, i segnali alti/bassi del rivelatore IR formano degli impulsi di differenti larghezze che corrispondono alle loro durate.



Messaggi dal Telecomando ad Infrarossi

Estratto da Infrarossi per il Controllo a Distanza del Boe-Bot

Gli esempi quì presentati si baseranno sul protocollo che i Telecomandi Universali usano per controllare gli apparecchi TV SONY®. Questo protocollo emette tredici lampeggi IR con una pausa tra ciascun impulso di circa mezzo millisecondo. Il risultato sono tredici impulsi negativi dal rivelatore IR che il BASIC Stamp può facilmente misurare. Il primo impulso è l'impluso di start che dura 2.4 ms. I successivi dodici impulsi dureranno 1.2 ms (1 binario) o 0.6 ms (0 binario). I primi sette impulsi dei dati contengono il messaggio IR che indica quale tasto è premuto. Gli ultimi cinque impulsi contengono un valore binario che specifica se il messaggio deve essere inviato ad un TV, VCR, CD, DVD player, etc. Gli impulsi sono trasmessi a partire dal bit meno significativo (sequenza LSB-first), quindi il primo impulso dei dati è il bit-0, l'impulso successivo è bit-1, e così via. Se premete e mantenete premuto un tasto sul telecomando, lo stesso messaggio verrà reinviato dopo una pausa di 20 - 30 ms.

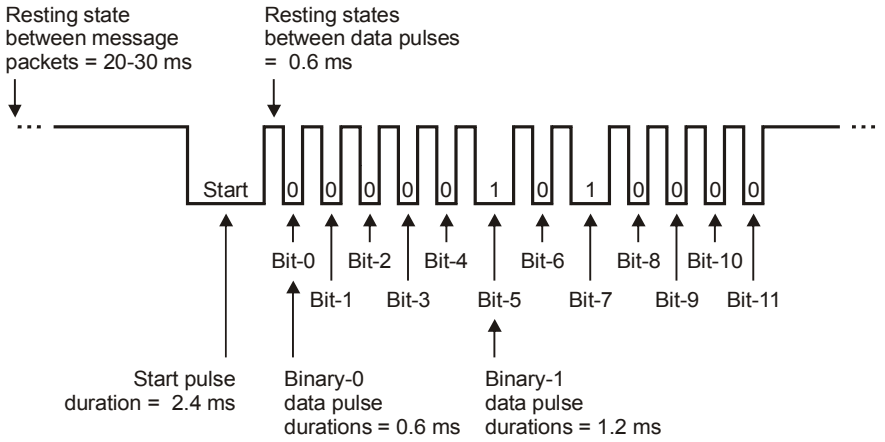
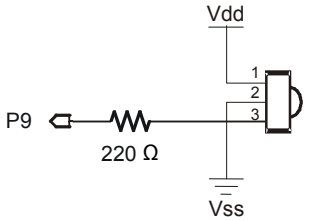


Diagramma di Temporizzazione IR

I valori sono approssimativi e possono variare leggermente da un telecomando all'altro.

Circuito di Rivelazione IR

Per le prove, tutto ciò che serve è questo circuito rivelatore IR ed il Terminale di Debug.



Circuit o di Rivelazione IR

Rivelatore IR visto dall'alto. Per la piedinatura vedere anche la Figura del Contenuto del Kit.

BASIC Stamp 2 Esempio "Basilare" – IrRemoteCodeCapture.bs2

Questo programma esempio spiega come acquisire e visualizzare un codice remoto con il BASIC Stamp 2. Se modificate la direttiva **`$STAMP`**, può anche essere usato con il BASIC Stamp 2e o 2pe.

- √ Assicuratevi di configurare il vostro telecomando universale per controllare un TV SONY®. Usate la documentazione allegata al vostro telecomando universale.
- √ Premere il tasto TV sul vostro telecomando in modo che sappiate di emettere segnali per TV.
- √ Scaricate o digitate e poi lanciate IrRemoteCodeCapture.bs2.
- √ Puntare il telecomando verso il rivelatore IR, e premere e rilasciare i tasti numerici.
- √ Provate anche i tasti POWER, CH+/-, VOL+/-, ed ENTER per vedere i codici di questi valori.

```
' Ir Remote Application - IrRemoteCodeCapture.bs2
' Process incoming SONY remote messages & display remote code.

' {$STAMP BS2}
' {$PBASIC 2.5}

' SONY TV IR remote variables

irPulse        VAR     Word                ' Stores pulse widths
remoteCode     VAR     Byte                ' Stores remote code

DEBUG "Press/release remote buttons..."

DO                                         ' Main DO...LOOP

  remoteCode = 0

  DO                                       ' Wait for end of resting state.
    RCTIME 9, 1, irPulse
  LOOP UNTIL irPulse > 1000

  PULSIN 9, 0, irPulse                     ' Get data pulses.
  IF irPulse > 500 THEN remoteCode.BIT0 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT1 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT4 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT5 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT6 = 1
```

```
  ' Map digit keys to actual values.
  IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
  IF (remoteCode = 10) THEN remoteCode = 0

  DEBUG CLS, ? remoteCode
LOOP                                          ' Repeat main DO...LOOP
```

Come Funziona IrRemoteCodeCapture.bs2

A ciascuna iterazione del ciclo **DO…LOOP** esterno, viene azzerato il valore di **remoteCode**. C'è anche un ciclo **DO…LOOP** interno con un comando **RCTIME** per rilevare la fine di un segnale alto maggiore di 2 ms. Questo indica che l'intervallo tra messaggi è appena terminato, e che stà cominciando l'impulso di start. Il primo comando **PULSIN** acquisisce il primo impulso dati, e la dichiarazione **IF…THEN** successiva usa il valore della variabile **irPulse** per impostare (porre ad 1 binario) o meno il bit corrispondente della variabile **remoteCode**. Dal momento che, mentre la dichiarazione **IF…THEN** è ancora in esecuzione, il secondo impulso dati è già iniziato, la durata residua del secondo impulso dati viene misurata con un comando **RCTIME**. Anche questo secondo valore è usato per impostare o meno il bit corrispondente di **remoteCode**. Tutto questo è ripetuto peer altre cinque volte per acquisire il resto della parte utile del messaggio IR ed impostare o meno i bit rimanenti di **remoteCode**.

I BS2sx e BS2p gestiscono i codici remoti in modo leggermente differente. I programmi normalmente ricercano l'effettivo impulso di start con un comando **PULSIN** invece di cercare l'intervallo tra messaggi. Essi usano anche comandi **PULSIN** per acquisire tutti gli impulsi visto che le dichiarazioni **IF…THEN,** che impostano i bit della variabile **remoteCode**, vengono ultimate prima del fronte di salita del secondo impulso dati. Per vedere un esempio di codice per i BS2sx, BS2p vedere la dichiarazione **#CASE** contenuta nella subroutine **Get_Ir_Remote_Code** del prossimo programma esempio.

Serie BASIC Stamp 2 Esempio Applicativo IrRemoteButtonDisplay.bs2

Potete usare questo esempio applicativo con i moduli BASIC Stamp 2, 2e, 2sx, 2p, o 2pe per provare il vostro telecomando e visualizzare quale tasto è stato premuto.

- √ Come per il precedente programma esempio,per prima cosa, assicuratevi che il vostro telecomando sia configurato per controllare un TV SONY.
- √ Aggiornate la direttiva **$STAMP** per il modulo BASIC Stamp che state usando.
- √ Scaricate o digitate e lanciate IrRemoteButtonDisplay.bs2.

- √ Puntate il telecomando verso il rivelatore IR, premere e rilasciare i tasti.
- √ Accertatevi che il Terminale di Debug riporti il tasto giusto. Iniziate con i numeri, canale, volume, etc.

Potete modificare od espandere la dichiarazione **SELECT…CASE** per provare i tasti VCR definiti nella sezione Costanti (Play, Stop, Rewind, etc.). Ci sono di solito diversi differenti codici per la configurazione dei telecomandi universali per controllare VCR SONY, quindi potreste dover fare un pò di prove prima di trovare il codice che "parli" lo stesso linguaggio PWM del controllore TV. Potete stabilire se il codice funziona perchè i tasti numerici, CH/VOL+/-, e POWER continueranno a funzionare dopo aver premuto il tasto VCR.

```
' -----[ Title ]-------------------------------------------------------------
' Ir Remote Application - IrRemoteButtonDisplay.bs2

' Process incoming SONY remote signals and display the corresponding button
' in the Debug Terminal.

' {$STAMP BS2}                              ' BS2, 2sx, 2e, 2p, or 2pe
' {$PBASIC 2.5}

' -----[ Revision History ]---------------------------------------------

' V1.0 - Supports most SONY TV and VCR control buttons.
'        Supports BASIC Stamp 2, 2SX, 2e, 2p, and 2pe modules.

' -----[ I/O Definitions ]----------------------------------------------

' SONY TV IR remote declaration - input receives from IR detector

IrDet          PIN     9                  ' I/O pin to IR detector output

' -----[ Constants ]---------------------------------------------------

' Pulse duration constants for SONY remote.

#SELECT $STAMP
  #CASE BS2, BS2E, BS2PE                  ' PULSE durations
    ThresholdStart CON 1000               ' Message rest vs. data rest
    ThresholdPulse CON 500                ' Binary 1 vs. 0 for PULSIN
    ThresholdEdge  CON 300                ' Binary 1 vs. 0 for RCTIME
  #CASE BS2P, BS2SX
    ThresholdStart CON 2400               ' Binary 1 vs. start pulse
    ThresholdPulse CON 500  * 5 / 2       ' Binary 1 vs. 0 for PULSIN
  #CASE #ELSE
    #ERROR This BASIC Stamp NOT supported.
#ENDSELECT
```

```
' SONY TV IR remote constants for non-keypad buttons

Enter          CON     11
ChUp           CON     16
ChDn           CON     17
VolUp          CON     18
VolDn          CON     19
Mute           CON     20
Power          CON     21
TvLast         CON     59                    ' AKA PREV CH

' SONY VCR IR remote constants

' IMPORTANT: Before you can make use of these constants, you must
' also follow the universal remote instructions to set your remote
' to control a SONY VCR.  Not all remote codes work,  so you may have to
' test several.

VcrStop        CON     24
VcrPause       CON     25
VcrPlay        CON     26
VcrRewind      CON     27
VcrFastForward CON     28
VcrRecord      CON     29

' Function keys

FnSleep        CON     54
FnMenu         CON     96

' -----[ Variables ]-------------------------------------------------------

' SONY TV IR remote variables

irPulse        VAR     Word                  ' Stores pulse widths
remoteCode     VAR     Byte                  ' Stores remote code

' -----[ Initialization ]--------------------------------------------------

DEBUG "Press/release remote buttons..."

' -----[ Main Routine ]----------------------------------------------------

' Replace this button testing DO...LOOP with your own code.

DO                                           ' Main DO...LOOP

  GOSUB Get_Ir_Remote_Code                   ' Call remote code subroutine

  DEBUG CLS, "Remote button: "               ' Heading
```

```
  SELECT remoteCode                          ' Select message to display
    CASE 0 TO 9
      DEBUG DEC remoteCode
    CASE Enter
      DEBUG "ENTER"
    CASE ChUp
      DEBUG "CH+"
    CASE ChDn
      DEBUG "CH-"
    CASE VolUp
      DEBUG "VOL+"
    CASE VolDn
      DEBUG "VOL-"
    CASE Mute
      DEBUG "MUTE"
    CASE Power
      DEBUG "POWER"
    CASE TvLast
      DEBUG "LAST"
    CASE ELSE
      DEBUG DEC remoteCode, " (unrecognized)"
  ENDSELECT

LOOP                                         ' Repeat main DO...LOOP

' -----[ Subroutine - Get_Ir_Remote_Code ]--------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0

  #SELECT $STAMP
    #CASE BS2, BS2E, BS2PE
      DO                                     ' Wait for end of resting state.
        RCTIME IrDet, 1, irPulse
      LOOP UNTIL irPulse > ThresholdStart
      PULSIN IrDet, 0, irPulse               ' Get data pulses.
      IF irPulse > ThresholdPulse THEN remoteCode.BIT0 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT1 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT2 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT3 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT4 = 1
      RCTIME IrDet, 0, irPulse
```

```
    IF irPulse > ThresholdEdge  THEN remoteCode.BIT5 = 1
    RCTIME IrDet, 0, irPulse
    IF irPulse > ThresholdEdge  THEN remoteCode.BIT6 = 1
  #CASE BS2SX, BS2P
    DO                                    ' Wait for start pulse.
      PULSIN IrDet, 0, irPulse
    LOOP UNTIL irPulse > ThresholdStart
    PULSIN IrDet, 0, irPulse              ' Get data pulses.
    IF irPulse > ThresholdPulse THEN remoteCode.BIT0 = 1
    PULSIN IrDet, 0, irPulse
    IF irPulse > ThresholdPulse THEN remoteCode.BIT1 = 1
    PULSIN IrDet, 0, irPulse
    IF irPulse > ThresholdPulse THEN remoteCode.BIT2 = 1
    PULSIN IrDet, 0, irPulse
    IF irPulse > ThresholdPulse THEN remoteCode.BIT3 = 1
    PULSIN IrDet, 0, irPulse
    IF irPulse > ThresholdPulse THEN remoteCode.BIT4 = 1
    PULSIN IrDet, 0, irPulse
    IF irPulse > ThresholdPulse THEN remoteCode.BIT5 = 1
    PULSIN IrDet, 0, irPulse
    IF irPulse > ThresholdPulse THEN remoteCode.BIT6 = 1
  #CASE #ELSE
    #ERROR "BASIC Stamp version not supported by this program."
#ENDSELECT

' Map digit keys to actual values.
IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
IF (remoteCode = 10) THEN remoteCode = 0

RETURN
```

Serie BASIC Stamp 2 Esempio - Applicazione Multi-Cifra

Potete usare la tastiera del telecomando per l'immissione di valori sostituendo il ciclo **DO…LOOP** nella routine principale di IrRemoteButtonDisplay.bs2 con quella mostrata sotto. Funziona per valori da 0 a 65535; digitate il valore sulla tastiera quindi premete il tasto ENTER del telecomando.

- √ Aggiungete questa dichiarazione ad IrRemoteButtonDisplay.bs2 nella sezione Variabili:

```
value          VAR     Word               ' Stores multi-digit value
```

- √ Sostituite con il programma mostrato sotto il ciclo **DO…LOOP** nella routine principale di IrRemoteButtonDisplay.bs2.
- √ Lanciate il programma e seguite le richieste del Terminale di Debug.

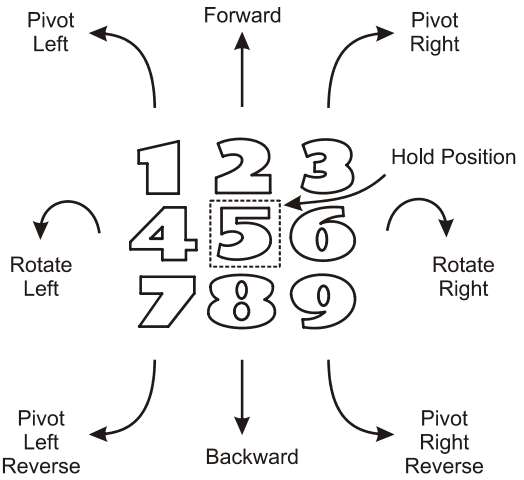
```
' Replace the DO...LOOP in the Main Routine with this one for multi-
' digit value acquisition (up to 65535). Value stored in value
' variable.

DEBUG CR, CR, "Type value from", CR, "0 to 65535,", CR,
      "then press ENTER", CR, CR

DO
  value = 0
  remoteCode = 0
  DO
    value = value * 10 + remoteCode
    DO
      GOSUB Get_Ir_Remote_Code
      IF (remoteCode > 9) AND (remoteCode <> Enter) THEN
        DEBUG "Use digit keys or ENTER", CR
        PAUSE 300
      ELSE
        DEBUG "You pressed: "
        IF remoteCode = Enter THEN
          DEBUG "Enter", CR
        ELSE
          DEBUG DEC remoteCode, CR
        ENDIF
        PAUSE 300
      ENDIF
    LOOP UNTIL (remoteCode < 10) OR (remoteCode = Enter)
  LOOP UNTIL (remoteCode = Enter)
  DEBUG ? value, CR, "Ready for next value...", CR
LOOP
```

Applicazione Boe-Bot per il BASIC Stamp 2

Questa nuova applicazione richiede un robot Boe-Bot con un modulo BASIC Stamp 2 che sarete in grado di controllare premendo e mantenendo premuti i tasti numerici per eseguire le manovre mostrate nella figura. Inoltre, potrete usare anche CH+ = avanti, CH- = indietro, VOL+ = rotazione a destra, VOL- = rotazione a sinistra.



Controllo della Direzione con la Tastiera Numerica

La seguente routine è per il robot Boe-Bot con i servo Parallax a rotazione continua. Il suo servo sinistro deve essere collegato a P13, ed il suo servo destro collegato a P12. se avete i servo PM Parallax, usate 500 invece di 650 e 100 al posto di 850 per gli argomenti ***Duration*** del comando **PULSOUT**.

√ Sostituite con questo codice mostrato sotto, il ciclo **DO…LOOP** della routine principale di IrRemoteButtonDisplay.bs2, lanciate il programma e guidate il Boe-Bot con il vostro telecomando. Divertitevi!

```
DEBUG CR, CR, "Press and hold digit", CR, "or CH+/-, VOL+/- keys", CR,
      "to control the Boe-Bot..."

DO
  GOSUB Get_Ir_Remote_Code
  SELECT remoteCode
    CASE 2, ChUp                              ' Forward
      PULSOUT 13, 850
      PULSOUT 12, 650
    CASE 4, VolDn                             ' Rotate left
      PULSOUT 13, 650
      PULSOUT 12, 650
    CASE 6, VolUp                             ' Rotate right
      PULSOUT 13, 850
      PULSOUT 12, 850
    CASE 8, ChDn                              ' Backward
      PULSOUT 13, 650
      PULSOUT 12, 850
    CASE 1                                    ' Pivot Fwd-left
```

```
      PULSOUT 13, 750
      PULSOUT 12, 650
    CASE 3                                  ' Pivot Fwd-right
      PULSOUT 13, 850
      PULSOUT 12, 750
    CASE 7                                  ' Pivot back-left
      PULSOUT 13, 750
      PULSOUT 12, 850
    CASE 9                                  ' Pivot back-right
      PULSOUT 13, 650
      PULSOUT 12, 750
    CASE ELSE                               ' Hold position
      PULSOUT 13, 750
      PULSOUT 12, 750
  ENDSELECT
LOOP
```

Ulteriori Risorse

Queste risorse sono disponibili presso www.parallax.com.

Lindsay, Andy. *IR Remote for the Boe-Bot,* Student Guide, Version 1.0, California: Parallax, Inc., 2004.
Questo libro è citato nella prima pagina di questi allegati al kit.

Williams, Jon. *The Nuts and Volts of the BASIC Stamps,* Volume 3, California: Parallax, Inc., 2003.
Column #76: *Control from the Couch* introduce l'acquisizione e la decodifica dei segnali IR di controllo dei TV SONY con il BASIC Stamp 2SX (o 2p).

Indice