Training Manual for the Tagisang Robotics 2014 Technical Training and Workshop

May 26-30, 2014

TRC 2014 Training Manual

Training Topic Schedule

Day1: Basic Electronics and Prototyping

Day 2: Gizduino Fundamentals

Day 3 - 4: Advanced Interfacing

Day 5: Mini Project

CONTENTS

Day 1.1: Basic Electronics

I. BASIC CONCEPTS - ELECTRICITY

A. Charge

- Basic unit from which electrical forces come from.

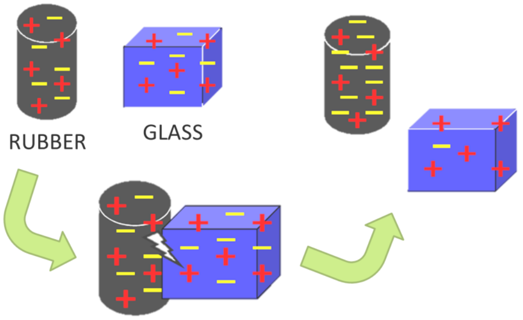
B. The Law of Conservation of Charge

"Charge can neither be created nor destroyed,

it can only be transferred from one body to another."

C. Static Electricity

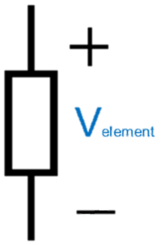
- When 2 electrically neutral objects are rubbed against each other, electrons can jump from one material to another.
- The amount of electrons that jump depends highly on the nature of the material.
- If the material loses electrons, it is now positively charged. If it gains electrons, it is now negatively charged
- When this phenomenon happens in a controlled environment, it produces the electricity we encounter and use everyday!



D. Voltage

- Potential Difference
- The work required to move a charge from one point to another
- The unit for voltage is volts

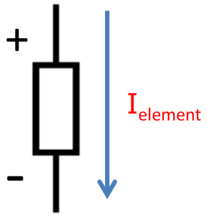
(V = joules/colomb)



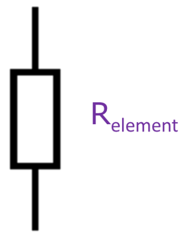
$V_{element}$ is the voltage **ACROSS** the element

E. Current

- The net flow or motion of charges
- DC – single direction AC – both direction (back and forth)



$I_{element}$ is the current **THROUGH** the element

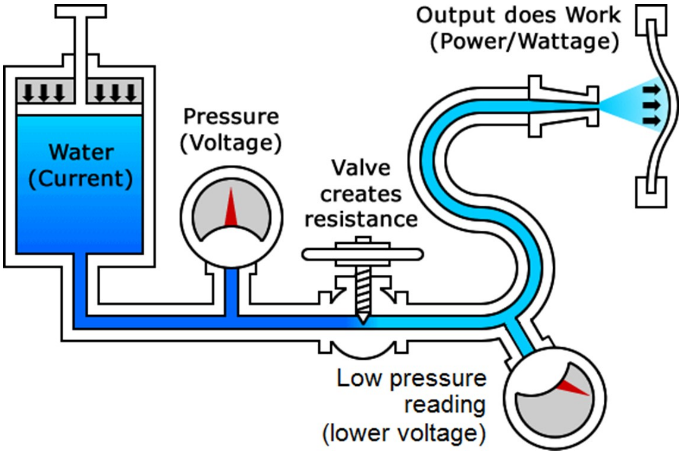


$R_{element}$ is the quantified resistance of the element

- Unit for current is amperes : (A = colomb/sec)

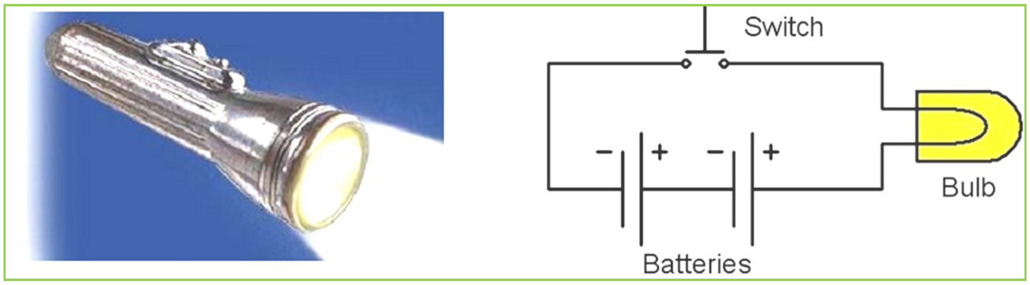
F. Resistance

- The ability of any material to limit the amount of current v passing through when a potential difference is applied
- Unit for resistance is ohms (Ω = volt/ampere)



G. Electrical Circuit

- Any arrangement of elements representing the 3 concepts that permits current to flow.



- Current flows once the circuit connection is closed
- Breaking the connection disrupts the current flow
- Batteries provide voltage
- Current flows, charges turn on bulb
- All elements present have various resistance values.

H. Ohm's Law

$$V = I \times R$$

"The current through an ideal conductor between two points is proportional to the potential difference across the two points."

I. Power Equation

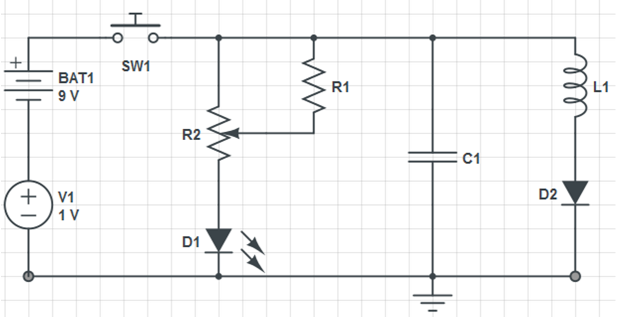
$$P = I \times V$$

- Equivalent to the work done per unit time
- Produced by the flow of charges (current) passing through an electric potential difference (voltage)

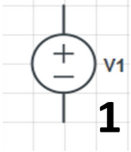
J. Water Analogy

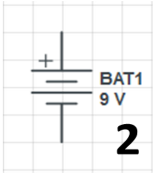
K. Circuit Diagram

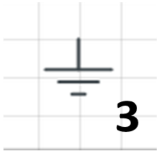
- Standard way of illustrating the components and connections in an electronics circuit
- Illustrated with the use of circuit element symbols

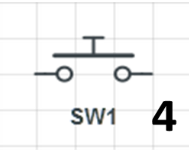


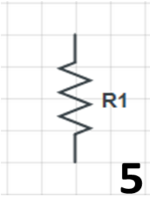
II. COMMON CIRCUIT ELEMENTS AND SYMBOLS

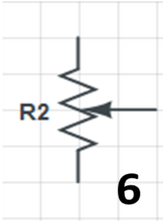


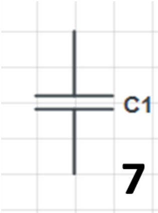


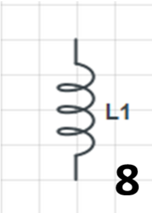


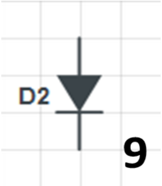


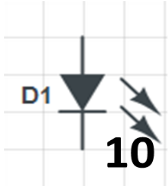




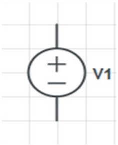


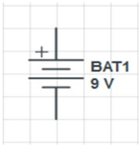






A. Voltage Source

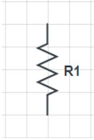


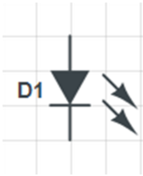


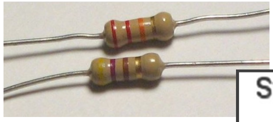
- Device that supplies a constant voltage
- By virtue of Ohm's Law, this device also supplies a relative amount of current

B. Resistor

- Current limiting component
- Colored bands determine the resistor's value within a range

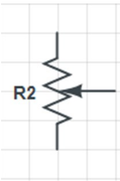






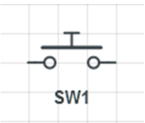
C. Potentiometer

- Resistors with variable sistance values
- Turning the knob changes the resistance of the component



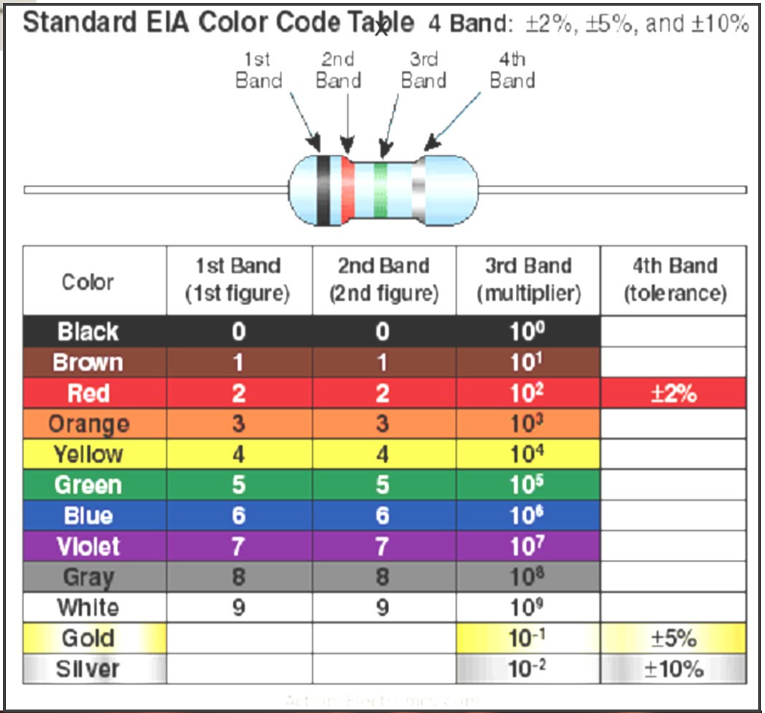
D. Switch

- Component that can open or close an electrical connection
- Changing of state is activated manually (button press)



E. Light Emitting Diode (LED)

- Keeps passage of current to one direction only
- Chemicals inside react differently to an applied voltage, producing color



Standard EIA Color Code Table 4 Band: ±2%, ±5%, and ±10%

1st Band
2nd Band
3rd Band
4th Band

Color	1st Band (1st figure)	2nd Band (2nd figure)	3rd Band (multiplier)	4th Band (tolerance)
Black	0	0	10^0	
Brown	1	1	10^1	
Red	2	2	10^2	±2%
Orange	3	3	10^3	
Yellow	4	4	10^4	
Green	5	5	10^5	
Blue	6	6	10^6	
Violet	7	7	10^7	
Gray	8	8	10^8	
White	9	9	10^9	
Gold			10^{-1}	±5%
Silver			10^{-2}	±10%

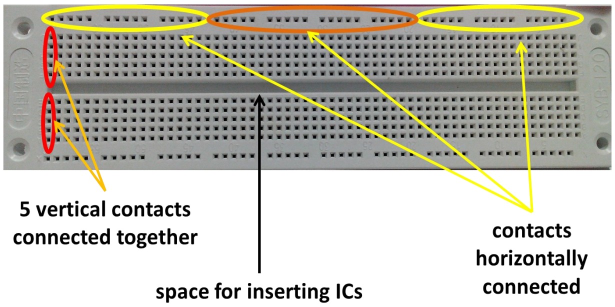
Day 1.2: Prototyping

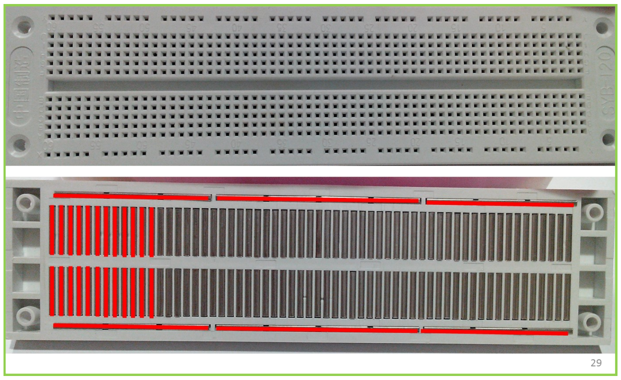
What we will be using

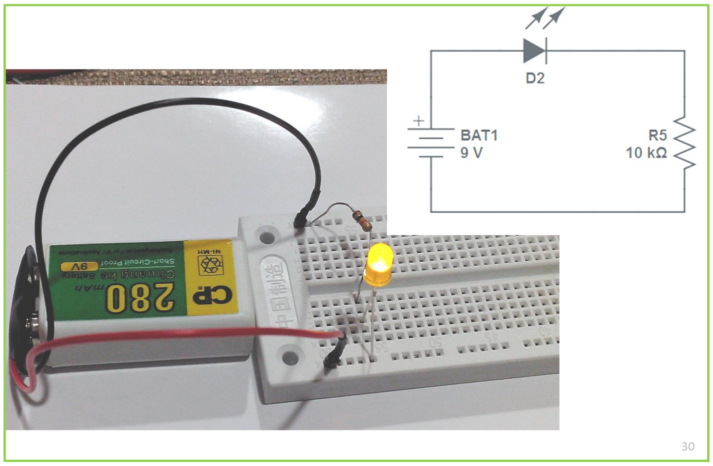
- Components (LEDs, Tact Switch, Resistors, Potentiometers)
- 9V Battery and battery clip
- Digital Multimeter
- Breadboard

I. Breadboard

- Construction base for prototyping electronics
- Also known as prototyping board or protoboard
- Used to make temporary circuits for testing
- Easy to change or replace connections/


5 vertical contacts connected together
space for inserting ICs
contacts horizontally connected



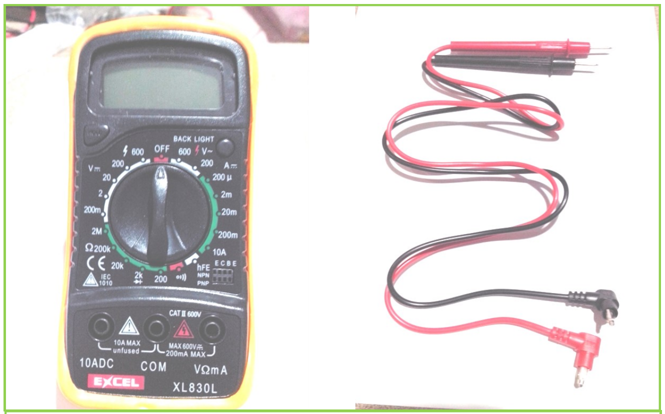


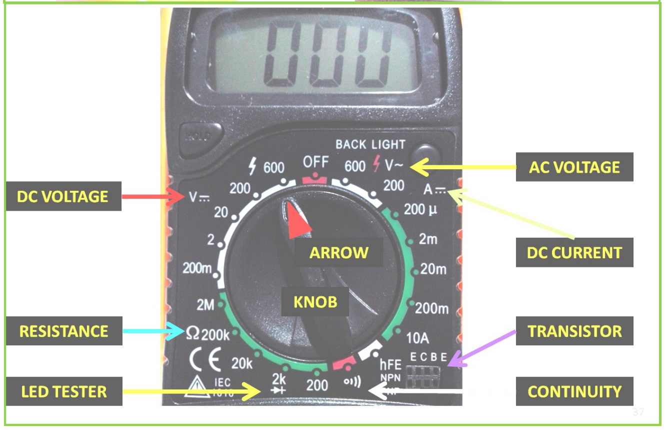
II. Breadboard Prototyping

- Disconnect the power supply from the breadboard before putting all the components
- Place the actual circuit component onto the breadboard one by one
- Always check if you are placing the correct component onto the breadboard
- Check connections between the circuit elements based on the circuit diagram

III. Digital Multimeter

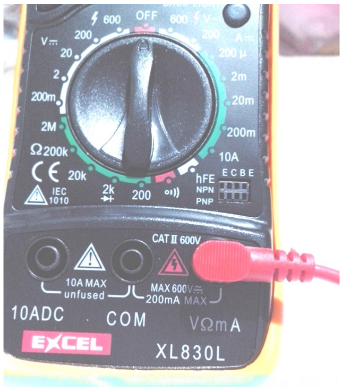
- A device used to measure/test the following:
  - DC Current
  - DC Voltage
  - AC Voltage
  - Resistance
  - Continuity
  - LED Tester

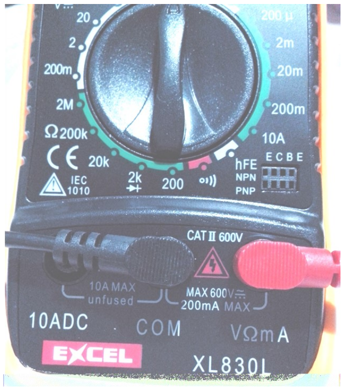




- RED Meter Lead
  - Voltage/Resitance or Amperage (VWmA) port
  - Positive Connection

- BLACK Meter Lead
  - Common ground / COM port
  - Negative Connection





- **PROBES**
  - Handles used to hold the tip on the connection being tested
- **TIPS**
  - End of the probe, provides the connection point

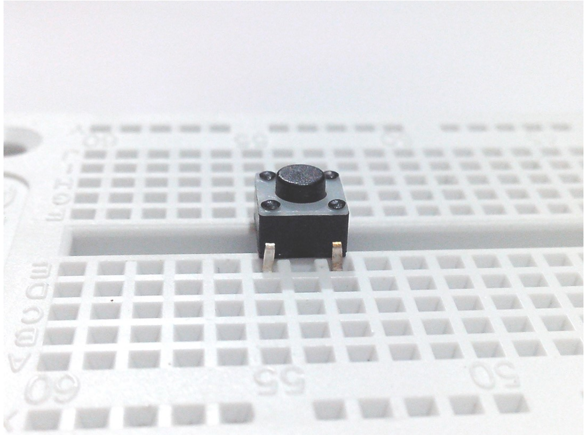
IV. Continuity Test

- Turn the knob and point the arrow to the continuity tester position
- Connect the tips of the black and red probes of the multitester, you should hear a an audible alarm when they are connected
- Disconnect power source first before performing continuity testing
- Connect the test leads across the wire / leads you want to check for continuity
- An audible alarm will be heard if they are connected, if not check the wire or the connection

WARNING!

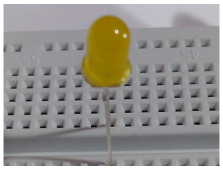
Do not use the continuity testing for power sources like batteries or power supplies

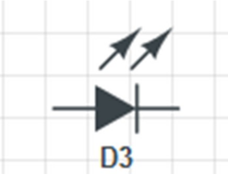
- Test the tact switch
  - Which leads are always connected?
  - Which leads are connected when you push the switch?



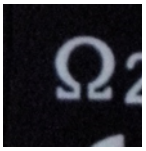
V. LED Test

- Turn the knob and point the arrow to the LED tester position
- LEDs will work one way only
- The longer wire goes to the positive
- The negative side is the shorter wire or the LED with the flat side
- Connect the RED meter lead to the positive side of the LED and the BLACK meter lead to the negative side of the LED
- LED will light up if the LED is working



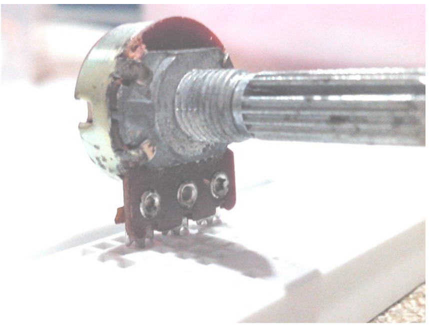


VI. Resistance Measurement



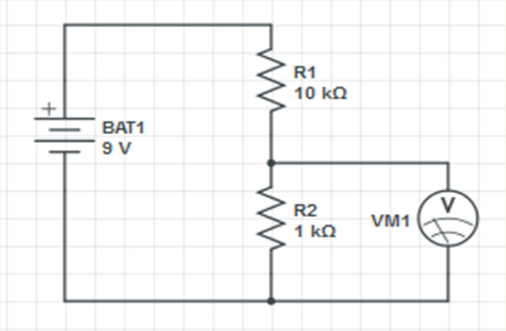
- Turn the knob and point the arrow to the Resistance Measurement position
- Disconnect any power source before testing
- Remove the component from the circuit before testing
- Set the dial to the lowest value.
- If OL or 1 appears, move to the next level
- Connect the test leads across terminals you want to measure resistance at
- DO NOT TEST POWER SOURCES!

- Get your potentiometer
- Measure resistance from the ff:



1. Both ends of the pot
2. Left end and middle of the pot
3. Right end and middle of the pot

TRY THIS!



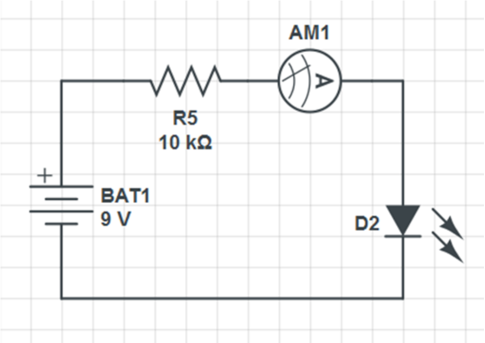
•Try turning the potentiometer knob, what happens?

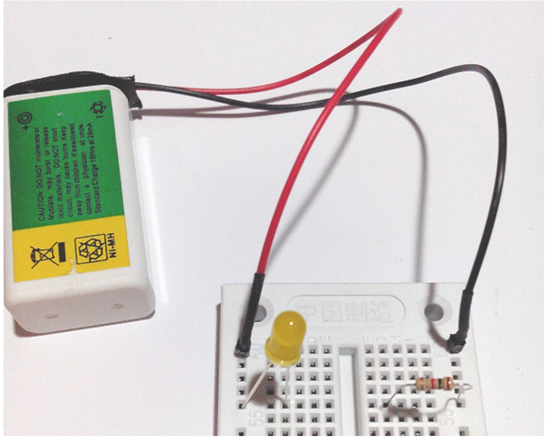
VII. DC Voltage Measurement

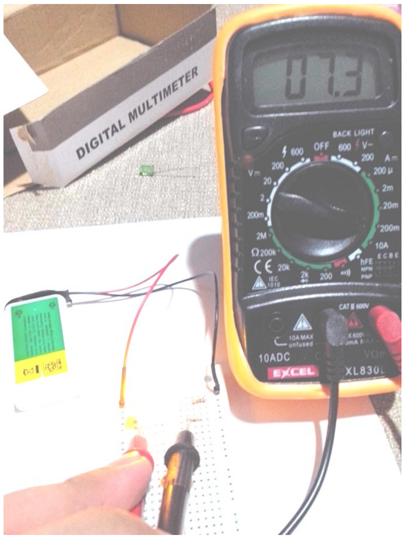
- Turn the knob and point the arrow to the DC Voltage Measurement position
- Use one value higher than your expected value.
- Ex. If you are measuring 9V, set the dial to 20V
- Connect the test leads across (in parallel) the terminals you want to measure the voltage at

VIII. DC Current Measurement

- Turn the knob and point the arrow to the DC Current Measurement position
- Use one value higher than your expected value.
- Ex. If you are measuring 9V, set the dial to 20V







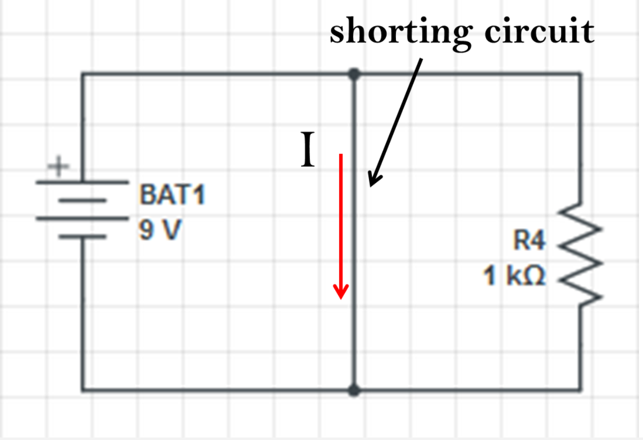
NOTES:

Day 1.3: Electrical Circuit Faults and Troubleshooting

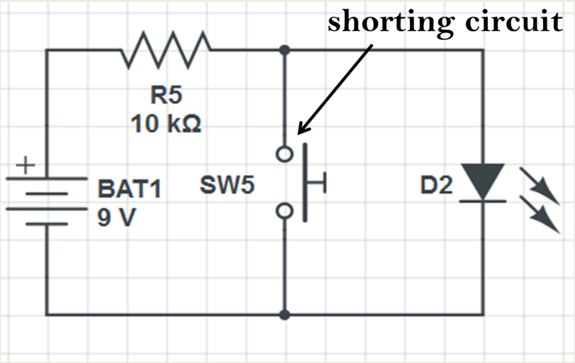
I. Short Circuit

- A low resistance connection established by accident or intention between two points in an electric circuit
- Current tends to flow through the path resistance is the lowest, bypassing the rest of the circuit
- Occurs as a result of improper wiring or broken insulation
- **TROUBLESHOOTING:**

- use continuity testing to check if there are accidental short circuits



TRY THIS!

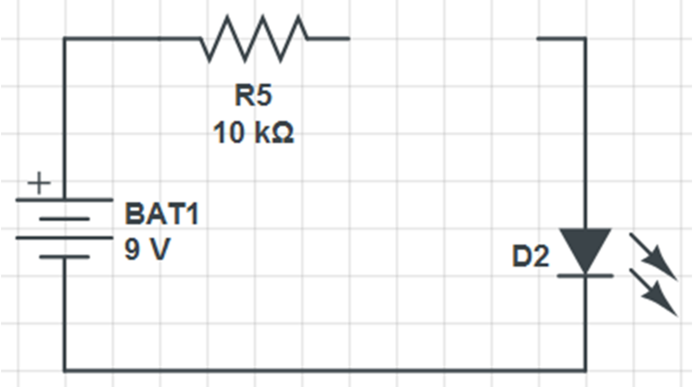


LED turns OFF when switch is pushed

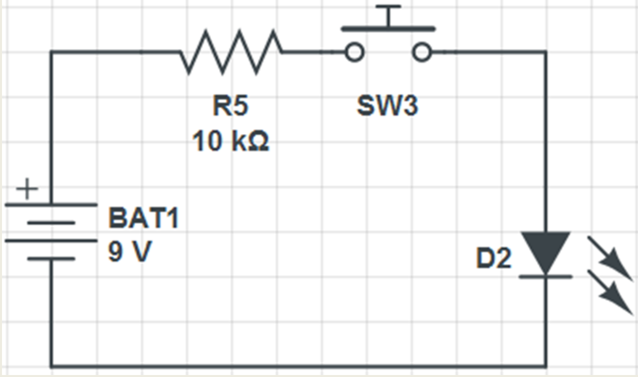
II. Open Circuit

- An electric circuit in which the normal path of current has been interrupted
- Accidental – disconnection of one part of the conducting pathway from another
- Intentional – by intervention of an electric component such as a switch
- **TROUBLESHOOTING:**

- use continuity testing to check if there are open circuits or unconnected elements



TRY THIS!



III. Reverse Connection/ Polarity

- Current flows only in one direction (DC circuits)
- Polarity of a source can be reversed by swapping the wires on the positive and negative terminal

(+) connects to (-)

(-) connects to (+)

- In DC circuits, make sure the polarity is correct

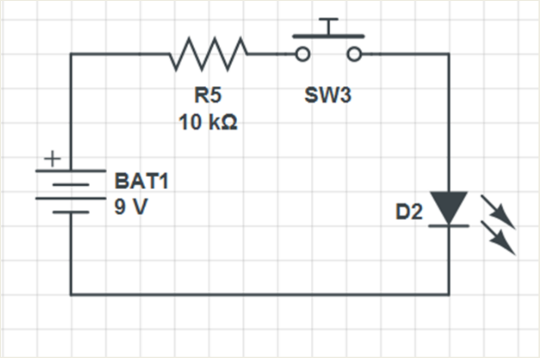
(+) to (+)

(-) to (-)

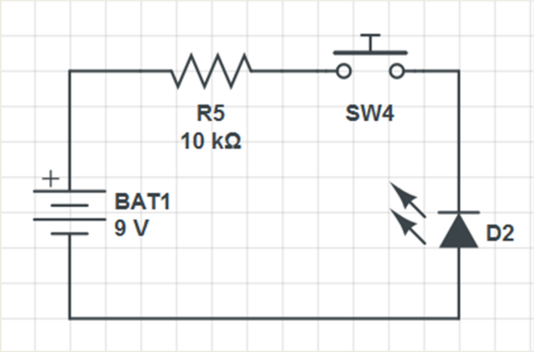
- **TROUBLESHOOTING:**

- Make sure that your polarity is correct before plugging in the power supply / voltage source

TRY THIS!



CORRECT



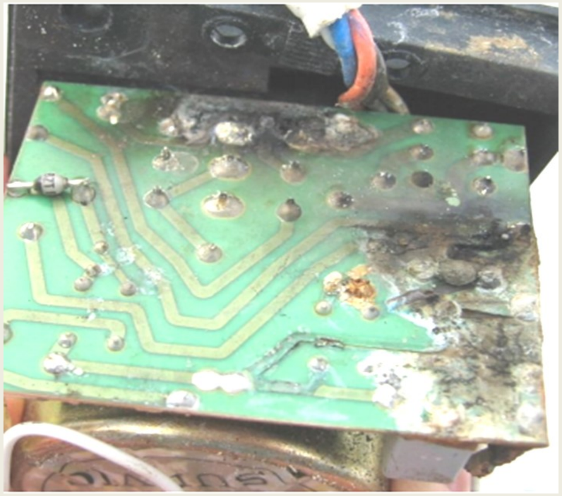
WRONG

ALWAYS REMEMBER

- Check all connections in your circuit before connecting it with the power source
- Check for accidental short and open circuits, loose connections
- Connect your voltage supply in the correct polarity

Verify if the components placed in the circuit are correct and have correct values

- Check also if there are defective components



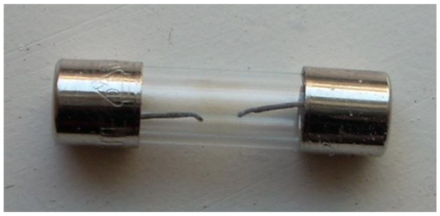
You do not want this to happen to you

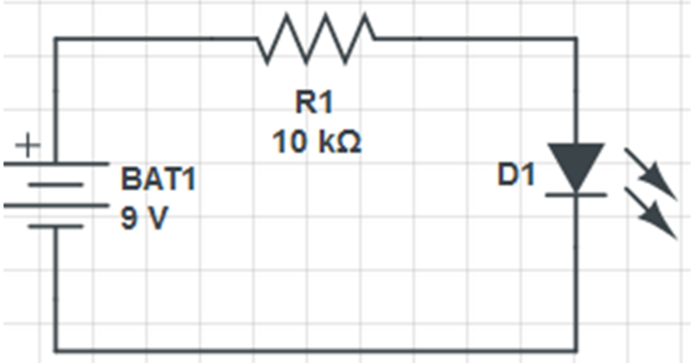
IV. Over Current

- A condition when the current in the circuit is larger than the intended current exists through a conductor
- May be caused by a short circuit, loose connection, excessive load, and incorrect design
- **TROUBLESHOOTING:**

- Check if there accidental short circuits, defective components and incorrect components / value

blown fuse due to overcurrent



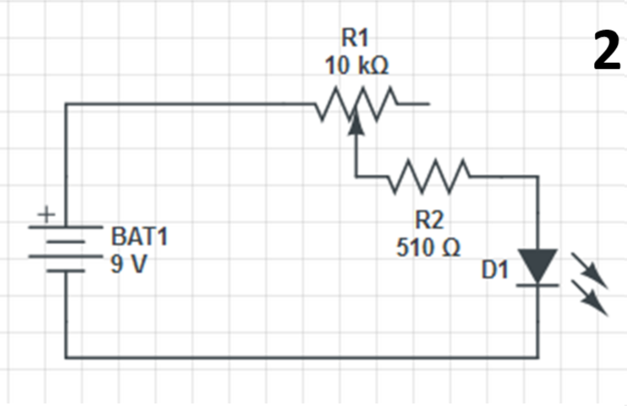


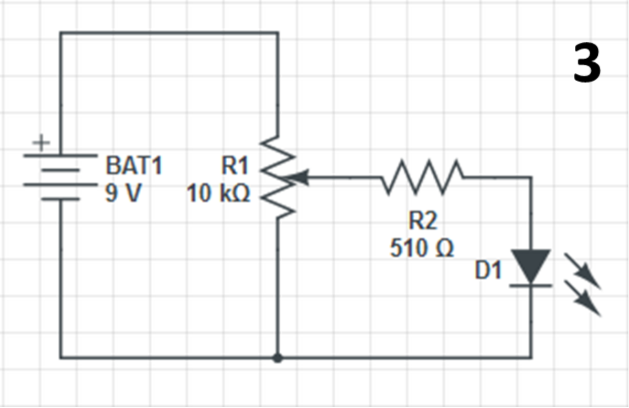
DAY 1.4: EXERCISES

EXERCISE 1

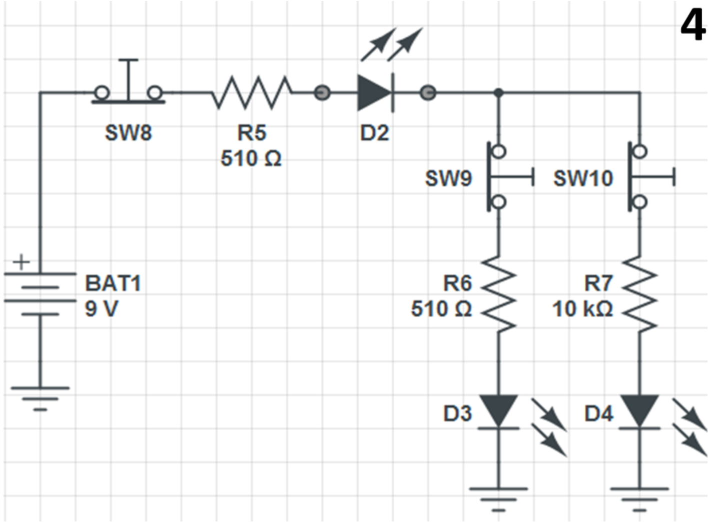
- Remove one leg of the resistor from the breadboard, what happens?
- Place a jumper wire across the LED, what hap-

DAY 1.4: EXERCISES

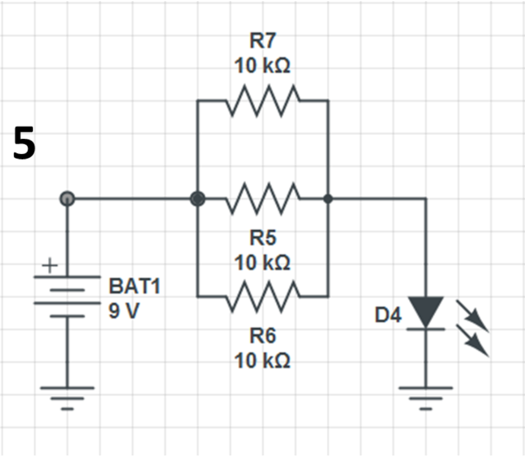


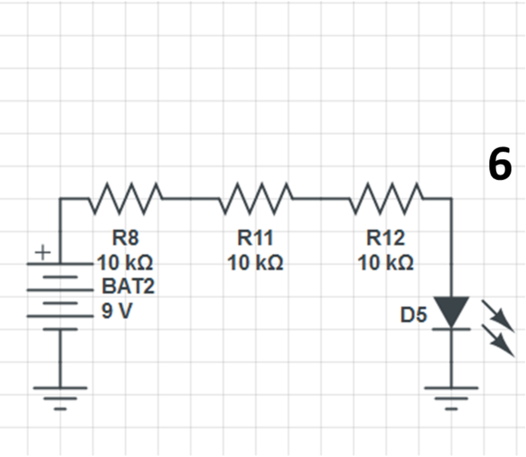


- Try turning the potentiometer knob to both sides, what happens to #2? How about #3?
- Why did the LED at #3 turned off when turning the pot to one side?



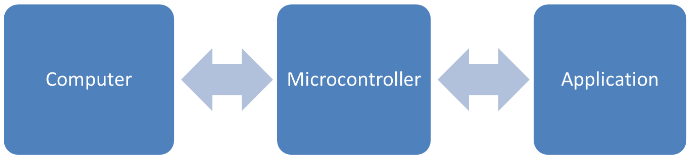
- Use jumper wires as your switches
- What happens to the LED when you remove the wire at A? at B? at C?
- Observe the light intensity of the LED in #5 and #6, which one is brighter? Why?



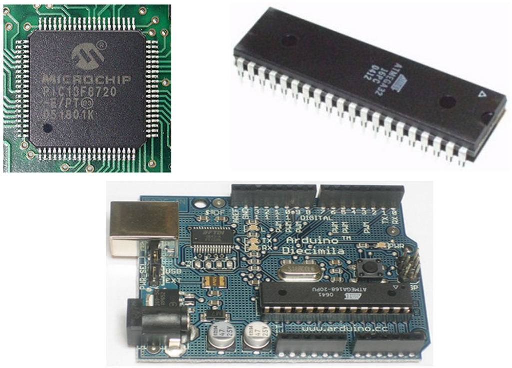


DAY 2: GIZDUINO FUNDAMENTALS

I. Introduction



A. What is a micronctroller?



- “One-chip solution”
- Highly integrated chip that includes all or most of the parts needed for a controller:
  - ∗ CPU
  - ∗ RAM
  - ∗ ROM
  - ∗ I/O Ports
  - ∗ Timer
  - ∗ Interrupt Controller

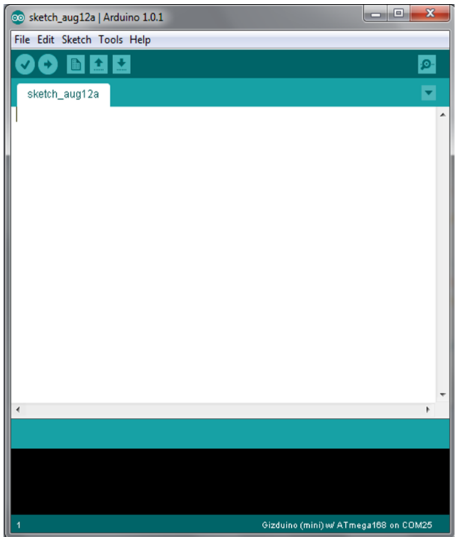
B. Microcontroller Applications

- Environmental Monitoring
- Automobiles
- Data Acquisition
- Toys
- Robotics

C. Arduino

- Arduino is an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.
- The Gizduino board, available at E-gizmo, is based on Arduino Diecimila, a microcontroller board based on ATMega164/324/644.
- Arduino comes with its own open-source Integrated Development Environment (IDE).
- Why Arduino?
  - ∗ Inexpensive
  - ∗ Cross-platform
  - ∗ Simple, clear programming environment
  - ∗ Open-source and extensible software
  - ∗ Open-source and extensible hardware

D. Integrated Development Environment

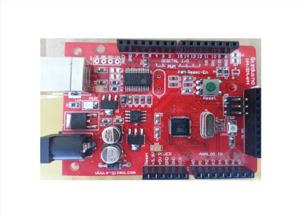

sketch_aug12a | Arduino 1.0.1
File Edit Sketch Tools Help
sketch_aug12a
1
Gizduino (mini) w/ ATmega168 on COM25

- It connects to the Arduino hardware to upload programs and communicate with them.
- Contains:
  - ∗ Toolbar
  - ∗ Text Editor
  - ∗ Status Bar
  - ∗ Console
  - ∗ Menus

II. Getting Started

Materials needed:

Gizduino Board



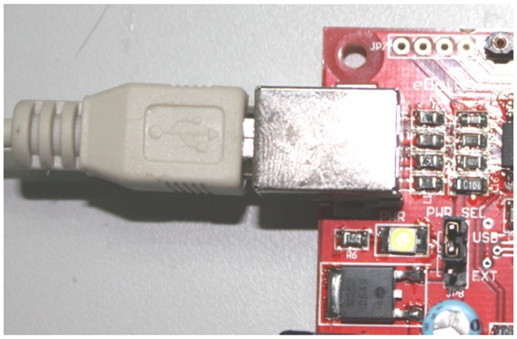
USB Cable

A.

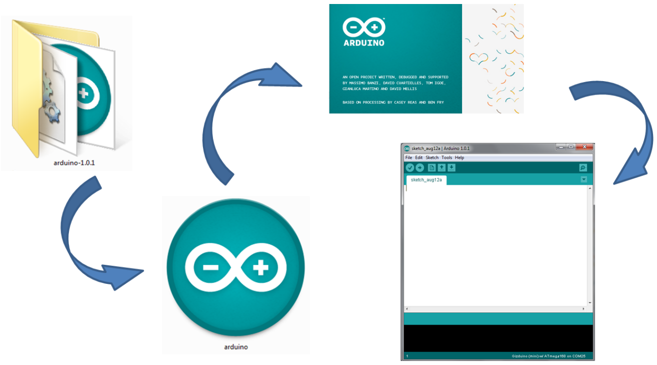
Connect Arduino to PC

- Connect the Gizduino board to you computer using the USB Cable. The green power LED should turn on.

B. Launching Arduino IDE



- Double click the Arduino application



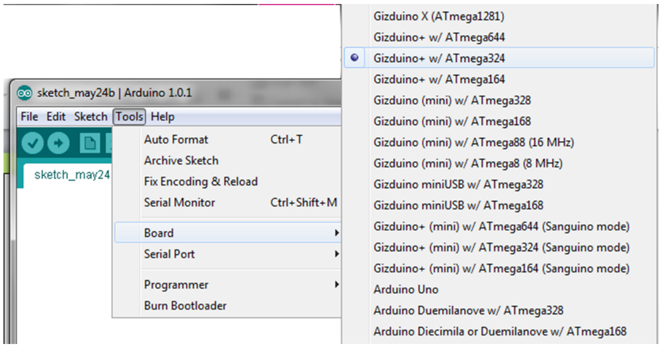
C. Creating a sketch

- Type the sketch below:

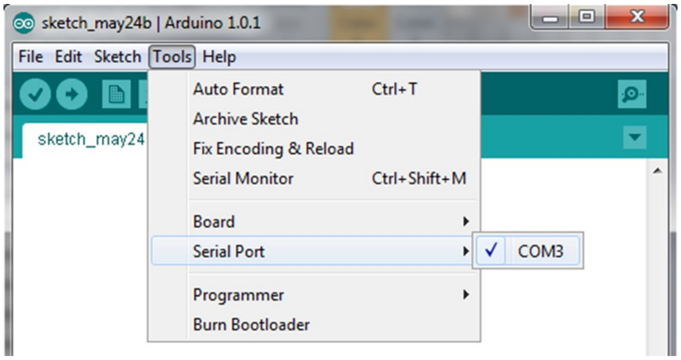
```
int ledPin = 13;
void setup(){
     pinMode(ledPin, OUTPUT); }
void loop(){
     digitalWrite(ledPin, HIGH);
     delay(1000);
     digitalWrite(ledPin, LOW);
     delay(1000); }
```

D. Setting up Arduino IDE

- Go to Tools -> Board menu and select Gizduino+ w/ ATmega644

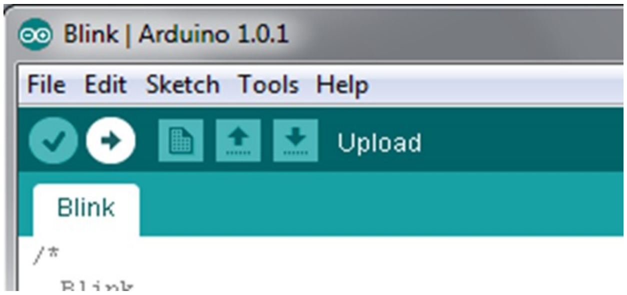

sketch_may24b | Arduino 1.0.1
File Edit Sketch Tools Help
sketch_may24
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
Gizduino X (ATmega1281)
Gizduino+ w/ ATmega644
Gizduino+ w/ ATmega324
Gizduino+ w/ ATmega164
Gizduino (mini) w/ ATmega328
Gizduino (mini) w/ ATmega168
Gizduino (mini) w/ ATmega88 (16 MHz)
Gizduino (mini) w/ ATmega8 (8 MHz)
Gizduino miniUSB w/ ATmega328
Gizduino miniUSB w/ ATmega168
Gizduino+ (mini) w/ ATmega644 (Sanguino mode)
Gizduino+ (mini) w/ ATmega324 (Sanguino mode)
Gizduino+ (mini) w/ ATmega164 (Sanguino mode)
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168

- Select the serial device of the Arduino board from the Tools -> Serial Port menu. To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.



E. Uploading the Program

- Save the sketch as Blink



- Click “Verify” button to check your code for syntax errors. After compiling, click “Upload” button in the environment. If the upload is successful, the message “Done Uploading” will appear in the status bar.

- A few seconds after the upload finishes, you should see pin13 (L) LED on the board start to blink (in orange).

  - If it does, congratulations! You’ve gotten Arduino up-and –running!

III. Sketch Fundamentals

We will now analyze the Blink code that we uploaded earlier. In this section, we will learn the following:

A. Sketch

B. Comments

C. Variables

D. Functions

E. Control Structures

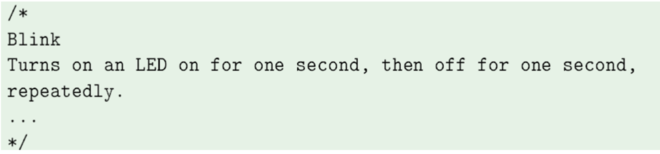
A. Sketch

- Sketch is the unit of code that is uploaded to and run on an Arduino board.
- Example:

  Blink (The sketch uploaded earlier)

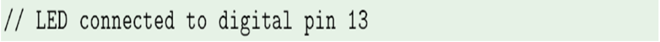
B. Comments

- Comments are ignored by the Arduino when it runs the sketch.
- It is there for people reading the code: to explain what the program does, how it works, or why it's written the way it is.
- For multi-line comment: use /* and */
- For single-line comment: use //
- Example
  - ∗ Multi-line comment:



```
/*
Blink
Turns on an LED on for one second, then off for one second,
repeatedly.
...
*/
```

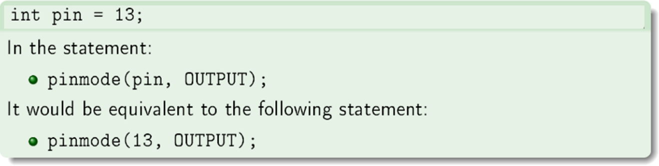
- ∗ Single line comment



```
// LED connected to digital pin 13
```

C. Variables

- Place for storing a piece of data
- Variable has a name, a value, and a type.
- Example



```
int pin = 13;
```

In the statement:

- `pinmode(pin, OUTPUT);`

It would be equivalent to the following statement:

- `pinmode(13, OUTPUT);`

D. Functions

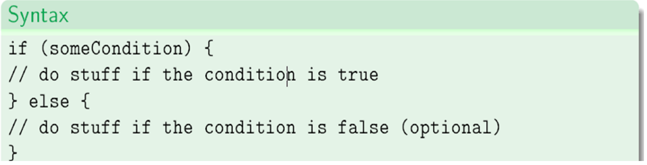
- Also known as procedure or subroutine.
- A named piece of code that can be used from elsewhere in a sketch.
- Special Functions

  There are two special functions that are part of every Arduino sketch:

  1. setup()
     - A function that is called once, when the sketch starts.
     - Setting pin modes or initializing libraries are placed here
  2. loop()
     - A function that is called over and over and is the heart of most sketches

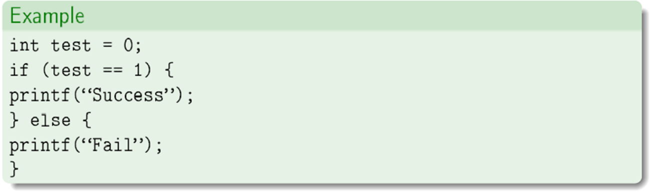
D. Control Structures

- If-Else Statement
  - ∗ Most basic of all programming control structures.
  - ∗ It allows you to make something happen or not depending on whether a given condition is true or not.



Syntax

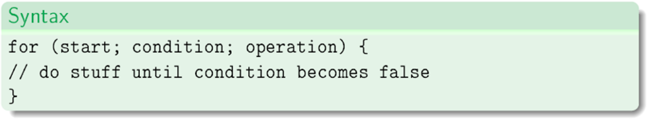
```
if (someCondition) {
// do stuff if the condition is true
} else {
// do stuff if the condition is false (optional)
}
```



Example

```
int test = 0;
if (test == 1) {
printf(“Success”);
} else {
printf(“Fail”);
}
```

- Loops
  - ∗ While Loop
  - ∗ Do-while Loop
  - ∗ **For Loop**

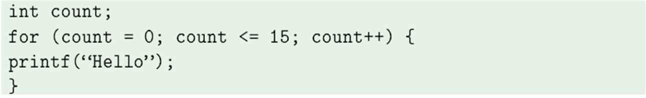


Syntax

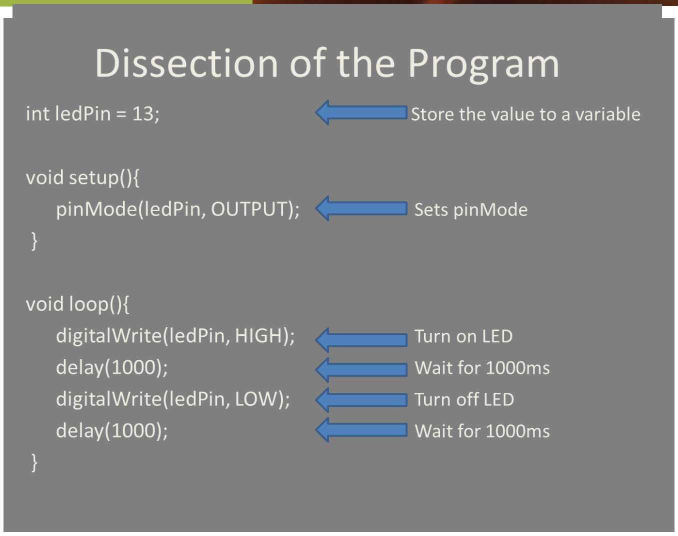
```
for (start; condition; operation) {
// do stuff until condition becomes false
}
```

Example:

- How many times will “Hello” be displayed?



```
int count;
for (count = 0; count <= 15; count++) {
printf(“Hello”);
}
```



Dissection of the Program

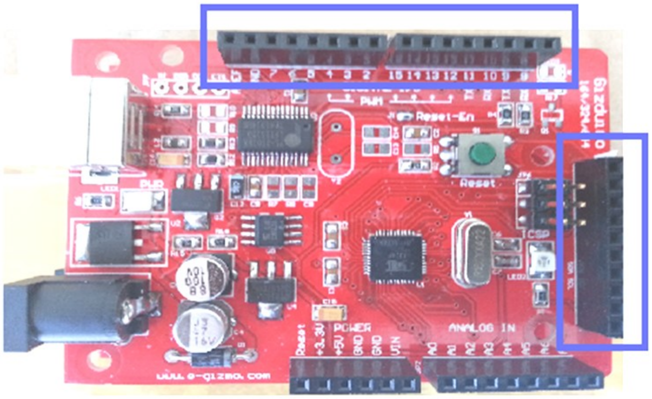
```
int ledPin = 13;                    Store the value to a variable

void setup(){
    pinMode(ledPin, OUTPUT);        Sets pinMode
 }

void loop(){
    digitalWrite(ledPin, HIGH);     Turn on LED
    delay(1000);                    Wait for 1000ms
    digitalWrite(ledPin, LOW);      Turn off LED
    delay(1000);                    Wait for 1000ms
 }
```

EXERCISE: VARYING BLINKING SPEED

- Change the value of the delay() to 200.
- Did the blinking speed of the LED's increased or decreased?
- Experiment on the function delay();
- Change them to any desired value. Observe what happens.

IV. DIGITAL INPUT/OUTPUT

- Digital = Discontinuous = Discrete
- Gizduino has 24 pins that can be configured as either digital input or output



A. "0" and "1"

- Logic "0"
  - ∗ Logic LOW
  - ∗ Ground (e.g. 0V)
  - ∗ False
- Logic "1"
  - ∗ Logic HIGH
  - ∗ Positive (e.g. 5V)
  - ∗ True

B. Setting Pin Mode

- pinMode()
- Configures the specified pin to behave either as an input or an output

Example
```
pinMode(13, OUTPUT); // Sets pin 13 as output
```

- Syntax:

  pinMode(pin, mode)

  - ∗ pin: the number of the pin whose mode you wish to set
  - ∗ mode: either INPUT or OUTPUT

Example
```
int ledPin = 13; // LED connected to digital pin 13
...
digitalWrite(ledPin, HIGH); // sets the LED on
delay();
digitalWrite(ledPin, LOW); // sets the LED off
delay();
```

C. Setting Pin Output

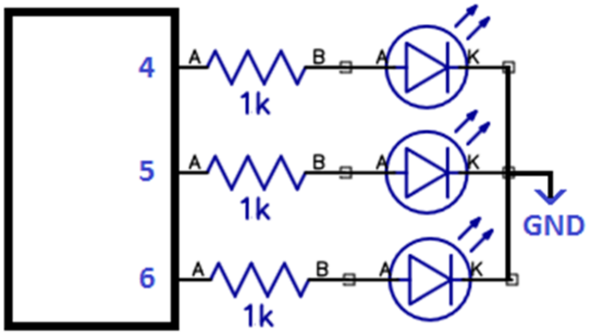
- digitalWrite()
- Write a HIGH or LOW value to a digital pin
- Syntax:

  digitalWrite(pin, value)

  - ∗ pin: pin number
  - ∗ value: HIGH or LOW

EXERCISE: BLINKING MULTIPLE LEDS

- Setup the circuit below on your breadboard. Modify your Blink code to turn ON and OFF three LEDs at the same time.
- Modify your code to turn ON and OFF one LED at a time



D. Reading Pin State

- digitalRead()
- Reads the value from a specified digital pin.
- Returns either HIGH or LOW
- Syntax:

  odigitalRead(pin)

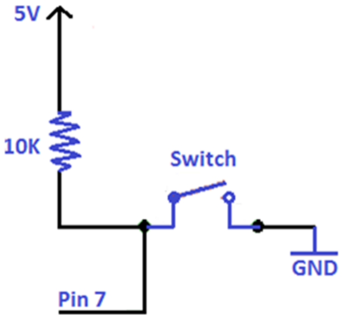
  ∗ pin: the number of the digital pin you want to read (int)

Example

```
int inPin = 7; // pushbutton connected to digital pin 7
...
val = digitalRead(inPin); // read the input pin
```

EXERCISE: PUSHBUTTON SWITCHING

- Setup a simple LED circuit. Use pin 4.
- Connect a switch as shown:
- Using the switch connected to pin 7 of Arduino, make a program that turns ON/OFF the LED.
- Button pressed = LED ON
- Button not pressed = LED OFF



CHALLENGE: LIGHTS ON, LIGHTS OFF

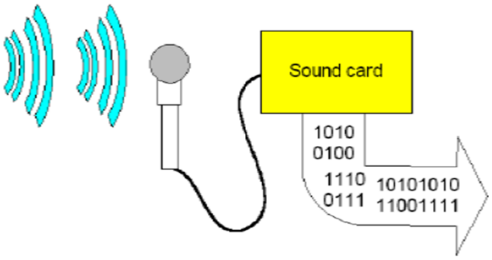
- Setup a simple LED circuit and connect a switch as with the previous exercise.
- Using the switch connected to pin 7 of Arduino, make a program that turns ON/OFF the LED.
  - ∗ Button pressed and released = LED ON
  - ∗ Button pressed and released again = LED OFF
  - ∗ Button pressed and released again = LED ON
  - ∗ And so on…

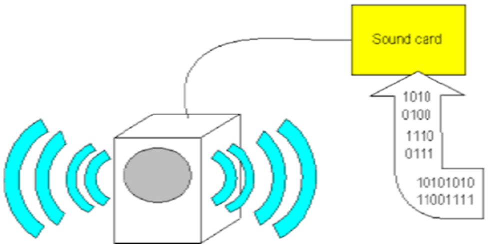
VI. ANALOG INPUT/OUTPUT

- We were able to make our microcontroller talk with our PC. But the world doesn't talk digital, only analog.

A. Analog to Digital Conversion

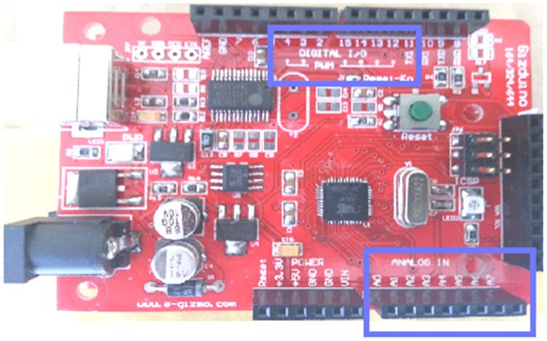
- Converts analog voltage signals from the real world to digital 1's and 0's that can be understood by the PC
- It is like translating a language.
- Analog Signal
  - ∗ Sound
  - ∗ Temperature
  - ∗ Light
- Digital Signal
  - ∗ 1's and 0's



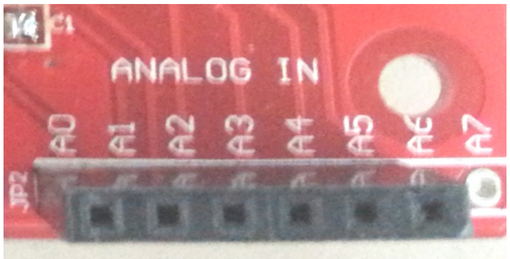


B. Analog Input/Output

- Analog = Continuous
- Gizduino has
  - ∗ 6 (labelled PWM out) analog output pins
  - ∗ 8 (labelled A0-A7) analog input pins



C. Analog Input



- These pins are for input usage only.
- You don't have to set the pinMode for these pins.

D. Reading Analog Input

- analogRead()
- Function to perform 10-bit ADC
- Returns value from 0-1023
- Syntax:

  analogRead(channel)

  Channel: ADC channel used

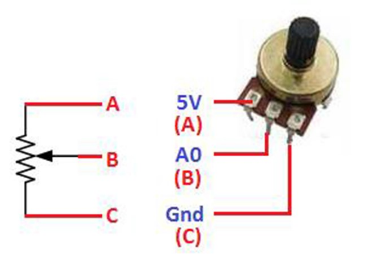
  Example: analogRead(0)

E. Analog Output

- You don't have to set the pinMode for these pins (3, 5, 6, 9, 10, 11) if you are using them as analog output pins.

analogRead Example

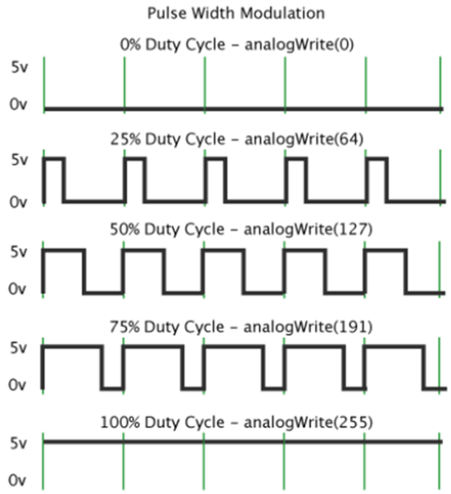
Connect a potentiometer to the Arduino as shown below



```
int baud = 9600;

int channel = 0;

void setup() {

    Serial.begin(baud);

}

void loop() {

    float sensorValue = analogRead(channel);
            // Read value from pin

    Serial.println(sensorValue, DEC);
            // Display in decimal format

    Serial.println(sensorValue*5/1024.0);       // Get
the voltage

    delay(1000);

}
```

- Open the Serial Monitor
- Turn the potentiometer knob
- What happens to the voltage when the knob is turned fully clockwise?
- What happens to the voltage when the knob is turned fully clockwise?
- What happens to the voltage when the knob is turned halfway across its full range?

- Pulse Width Modulation



F. Writing Analog Output

- analogWrite()
- Supported pins: 3, 5, 6, 9, 10, 11
- Function to perform 8-bit DAC
- Value ranges from 0-255 (0-5 volts counterpart)
- Syntax:

  analogWrite(pin, value)

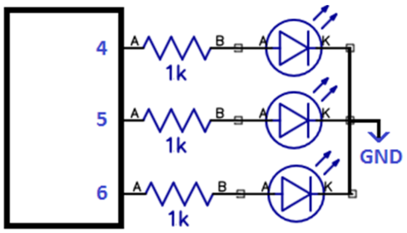
  Pin: the pin to write to

EXERCISE: PULSE WIDTH MODULATION

- Create a sketch that changes the brightness of an LED by turning the knob on a potentiometer.
- LED - any PWM pin and GND
- Potentiometer wiper to A0
- Potentiometer outer pins to 5V and GND

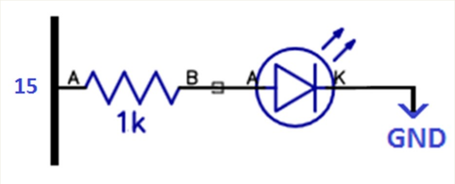
CHALLENGE: POTENTIOMETER CONTROL TO BLINKING LEDS

- Construct the LED setup shown below
- LED1 On if pot is set fully clockwise.
- LED2 On if pot is set halfway.
- LED3 On if pot is set fully counter clockwise.
- Otherwise, all LEDs are turned OFF



analogWrite Example

Setup an LED circuit in your breadboard. Connect it to ANY PWM PIN.



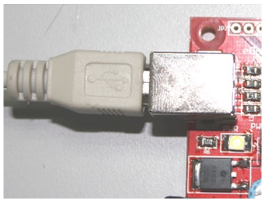
```
int ledPin = 15;  // OR ANY PWM PIN

void setup() {

}

void loop() {

     // fade in from min to max in increments of 5 points:

     for(int fadeValue = 0 ; fadeValue <= 255; fadeValue
+=5) {

          analogWrite(ledPin, fadeValue);                    //
sets value (0 to 255):

          delay(30); }

     // fade out from max to min in increments of 5 points:

     for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -
=5) {

          analogWrite(ledPin, fadeValue);                    //
sets value (0 to 255):

          delay(30);

}                         // What is performed in this sketch?
```

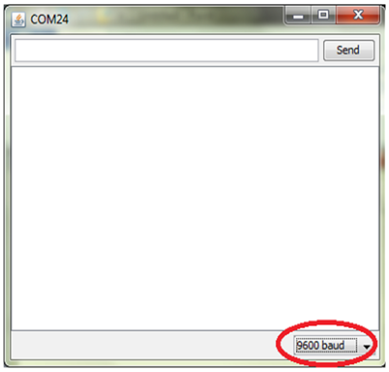
NOTES:

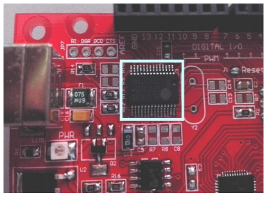
DAY 3 & 4: ADVANCED INTERFACING

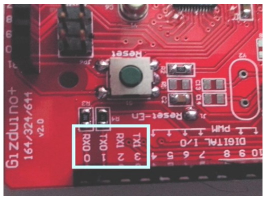
I. Serial Communications

- How it "talks" with your PC
  - A. USB cable
  - B. PL2303 chip
  - C. Serial pins









- The Gizduino communicates serially
  - Serial = "one after another"
  - Stream of 1's and 0's

A. **UART (Universal Asynchronous Receiver/ Transmitter**

- Main communications device employed on the board
- Accompanies the RS232 communications standard, made available to almost all electronic modules with a communications interface

B. Initialize Serial communications

- Serial.begin(<baud>);
- Need to do this ONCE, usually first command in setup();
- The value <baud> specifies the baud rate to be used (e.g. 9600, 38400, 115200, etc)
- **IMPORTANT: baud rates between communicating devices MUST MATCH!**

C. Pass data to the Serial port

- Serial.println(<value>);
- Can also use Serial.print();
- Able to display any value or data, represented by <value>
- **IMPORTANT: for CHAR enclose <value> in single quotes, for STRING enclose in double quotes!**

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(“Hello World!”);
  delay(1000);
}
```

D.

Poll the Serial line for incoming data

- variable = Serial.available();
- Returns the number of bytes to be read (int data type)
- **IMPORTANT: leave the inside of the parenthesis BLANK!**

E. Read the first byte of incoming data

- variable = Serial.read();
- Need to store the read data inside a variable (int or char data type)
- **IMPORTANT: data is read byte by byte or ONE CHARACTER PER READ COMMAND, needs extra processing if expected data are STRINGS**

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    char ReceivedByte = Serial.read();
    Serial.print(ReceivedByte);
    delay(10);
  }
}
```

NOTES:

____

____

____

____

____

____

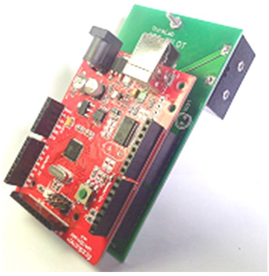
____

II.

TRC Pilot Board and Controller

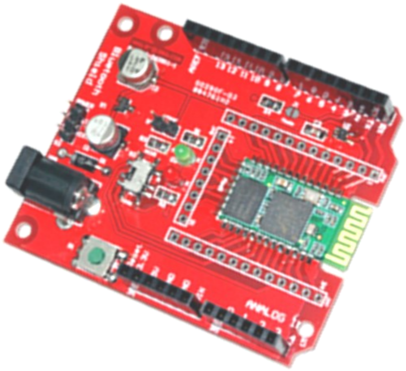
INPUT	ACCESS
Left stick Y-axis	Analog Pin 1
Left stick X-axis	Analog Pin 2
Right stick Y-axis	Analog Pin 3
Right stick X-axis	Analog Pin 4
UP button	Digital Pin 9
LEFT button	Digital Pin 10
DOWN button	Digital Pin 11
RIGHT button	Digital Pin 12
1 button	Digital Pin 5
2 button	Digital Pin 6
3 button	Digital Pin 7
4 button	Digital Pin 8
L1	Digital Pin 3
L2	Digital Pin 4
R1	Analog Pin 5
R2	Digital Pin 2

- While docked, upload the pilot_board_debug sketch to your Gizduino
- Open the Serial Monitor to view the outputs
- NOTE: Make sure the Serial Monitor baud rate is set at 38400!



II. Communications via Bluetooth

A. Communications



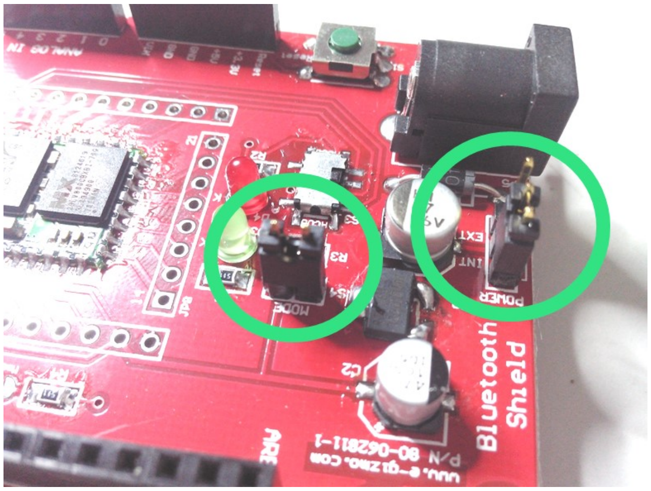
- Wired Medium
    - Serial/ UART
    - SPI
    - I2C
- Wireless Medium
    - GSM/ Mobile
    - WiFi
    - ZigBee/RF
    - **BLUETOOTH**

B. Bluetooth

- Transmits wirelessly at low power (~1 mW)
- Distance is limited to 10m radius
- Uses spread-spectrum frequency hopping

C. Configuring Bluetooth

- Unpower all boards
- Set POWER to INT, short MODE pins



- Using jumper wires, connect the following pins:

GIZDUINO	BT SHIELD
Pin 10	RX Pin
Pin 11	TX Pin
+5V	+5V
GND	GND

- Upload BTconfig sketch for Bluetooth Shield configuration
- Open the Serial Monitor and set the baud rate to 38400, also make sure the return line is set to Both NL & CR
- Type AT on the reply box. The Bluetooth device should reply and print OK

ROLE

- for MASTER, type: AT+ROLE=1
- for SLAVE, type: AT+ROLE=0

BAUD RATE

- AT+UART=38400,0,0

PAIRING KEY

- AT+PSWD=xxxx

Where xxxx represents a 4-digit number

SLAVE ADDRESS

- AT+ADDR

If successful, device will reply with a colon-separated digit string, ex. 2013:3:40964

BIND TO MASTER

- AT+BIND=<value>

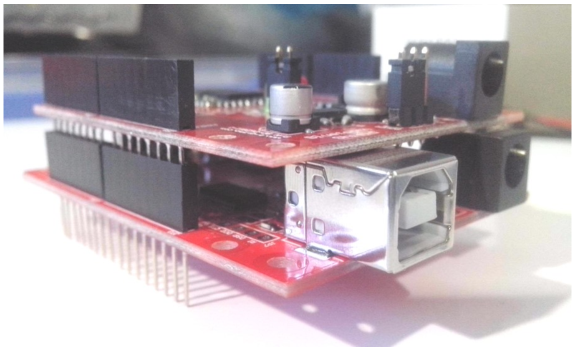
Where <value> is equal to your slave address, changing ALL COLONS to COMMAS (ex. 2013,3,40964)

D. Deploying Bluetooth

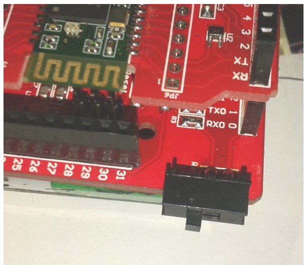
- Close Serial Monitor, unplug USB, and remove MODE jumper
- Plug again the USB, pairs should connect after a few seconds (RED LED LIGHTS UP!)
- Open the Serial Monitor to communicate with your paired device, the connection should be two-way

EXERCISE

- Fix the Bluetooth Shield and Gizduino connection by mounting them like so



- Move the Serial Comms switch to the left, as seen here.



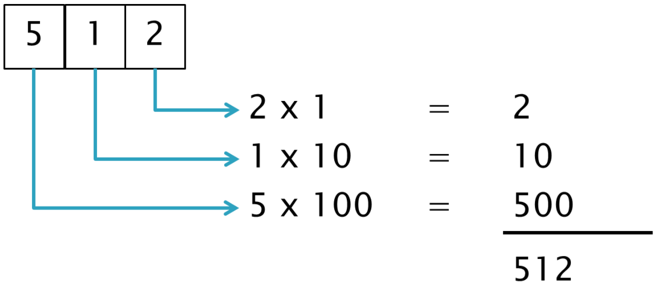
- Upload the BT_BasicControl sample sketch to your Gizduino
- Unplug USB, return the Serial Comms switch to the right
- Mount one of the Gizduinos with the Bluetooth Shield back to the controller and power it with a 9V battery
- On the other Gizduino, connect an LED on Pin 13 then plug it on the USB connector to power it up

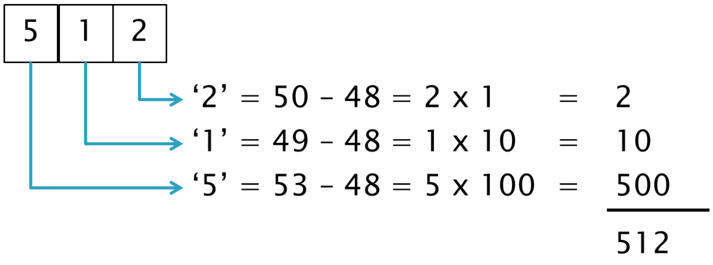
E. Character Translation

- ASCII characters have their equivalent decimal values

ASCII	Value	ASCII	Value
‘0’	48	‘5’	53
‘1’	49	‘6’	54
‘2’	50	‘7’	55
‘3’	51	‘8’	56
‘4’	52	‘9’	57

- Concept used in translating is base counting
- Set a terminating character to indicate when the program will stop processing
- While terminating character is not encountered, assume that the number being passed ends with

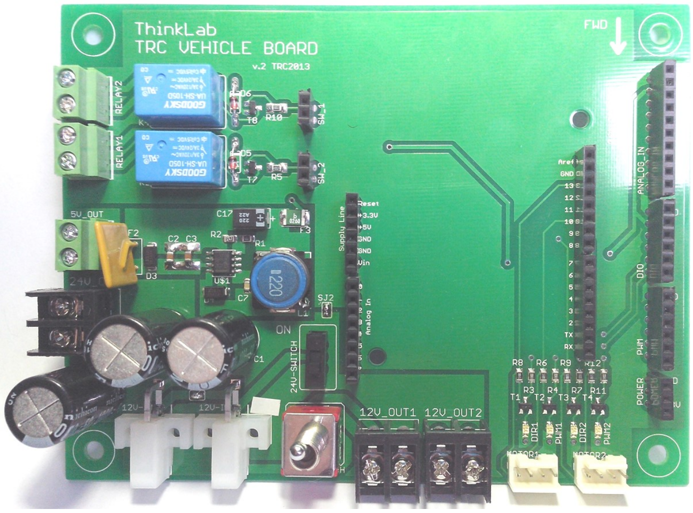


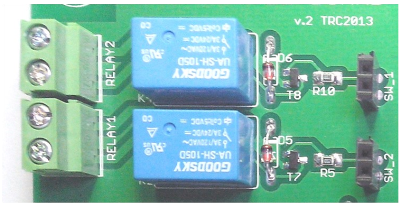

5
1
2
'2' = 50 – 48 = 2 x 1 = 2
'1' = 49 – 48 = 1 x 10 = 10
'5' = 53 – 48 = 5 x 100 = 500
512

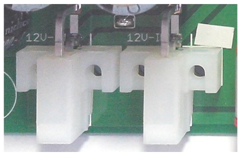
the current digit

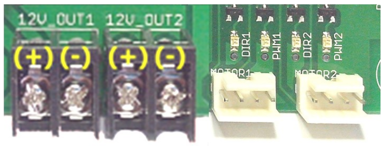
III. TRC 2013 Vehicle Board

- RULE 1 – Never place the Vehicle Board on/near any metallic or conductive surface/material
- RULE 2 – Make sure battery cable connectors are connected to the battery in proper polarity









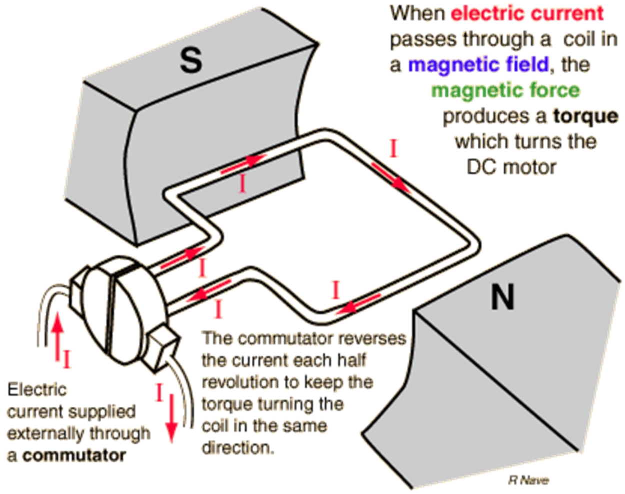
- RULE 3 – Always terminate all connections to the Vehicle Board in the proper polarity (GND goes to GND, supply goes to supply)

IV. Mechatronics: Servo and DC Motors

A. DC Motor

B. Tricky to connect to microcontrollers!

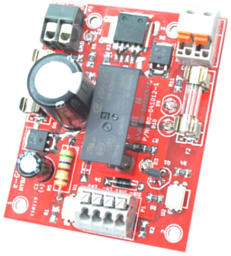
- Needs a lot of power to run
- Produces lots of electrical noise that may interfere with microcontroller pin outputs


When electric current passes through a coil in a magnetic field, the magnetic force produces a torque which turns the DC motor
S
N
I
I
I
I
I
I
I
The commutator reverses the current each half revolution to keep the torque turning the coil in the same direction.
Electric current supplied externally through a commutator
R Nave

- Provide separate power source for the DC Motor – NEVER POWER IT STRAIGHT FROM GIZDUINO PINS or +5V!
- Ensure all ground points (GND) are connected (common)

B. DC Motor Control

- SPEED
  - ∗ Pulse Width Modulation (PWM)
  - ∗ Supported by Gizduino pins 3, 5, 6, 9, 10, 11
  - ∗ Recall: analogWrite();
- DIRECTION
  - ∗ Motor Direction follows Source Current Direction
  - ∗ Needs external circuitry that will reverse the direction of the current coming from the motor's power supply
  - ∗ Direction is binary logic!
- Numerous ways to implement this control scheme!
  - ∗ Collective term – DC Motor Driver
  - ∗ H-Bridge configuration is the most commonly used and simplest to build



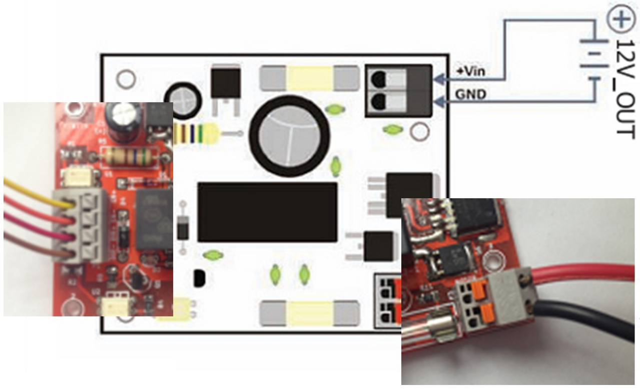
- Hybrid Driver

  ∗ Electro-mechanical solution for controlling DC Motor speed and direction

  ∗ Rugged design allows high current use, with additional safety features such as fused inputs and isolated microcontroller connections

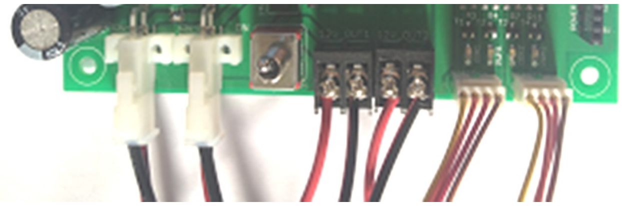
C. Hybrid_RunTest sample sketch

- Upload Hybrid_RunTest sample sketch to your Gizduino
- Completely wire the minimum set of connections to the Vehicle Board. Dock your Gizduino, and turn all switches ON.



D. Servo Motor

- Composed of an electric motor mechanically linked to a potentiometer interfaced with a control chip



- Signals are translated into position commands

- Servo motors usually has three wires for power, ground, and control
- Supported by Servo library for easy interfacing with Arduino system

SERVO	GIZDUINO
Red	+5V
Brown	GND
Yellow	Gizduino Pin

- Sketch > ImportLibrary > Servo

E. Servo Motor Control

- Create a Servo object

  Servo <name>;

  ∗ Declare Servo presence to Gizduino

  ∗ Identified by the <name> parameter, give your Servo a name/label

- Attach the servo to a Gizduino pin

  <name>.attach(pin);

  ∗ Bind the Servo to a specific pin on the Gizduino, pin will provide the rotation data for the attached Servo

  ∗ All DIO pins are supported, able to control multiple Servo motors

- Set angle of rotation

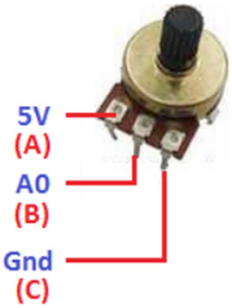
  <name>.write(angle);

  ∗ Rotates the shaft based on the specified value angle, between 0 to 180 only

  ∗ Position is held until a new value angle is given, or the power to the Servo is reset

  ∗ Use for loops to count from 0 to 180 and back

  ∗ Reconnect your potentiometer and use it to control your Servo, make Sevo follow the rotation of the potentiometer knob



F. Servo Motor Tips

- Power consumption of the Servo is directly proportional to its load, that is, the heavier the load the more current it draws
- All Servos are rated with a Stall Torque value, or the maximum load it can carry at a given operating voltage
- Displacement of the load increases as it is placed farther away from the shaft on the Servo arm

CHALLENGE: – SERVO CONTROL VIA BLUETOOTH

- }Using the right controller joystick, move the Servo from 90 to 0 when moving the joystick UP, and move from 90 to 180 when moving the joystick DOWN
- EXTRA CHALLENGE! – Control TWO Servos by mapping the other Servo to the left-right movement of the joystick

Tagisang Robotics 2014 – Training Kit Parts List	
Gizduino	1
USB cable connector	1
Bluetooth Shield	1
9V rechargeable battery	2
9V battery charger	1
Breadboard	1
Assorted Resistors	30
Tact Switches	5
LEDs	10
Wire Conn Male-Male single 20mm	20
Wire Conn Male-Male single 40mm	20
Potentiometer	4
Servo motor standard	2
Toolbox	1
User Manual	1

NOTES: