Kuwait University

Department of Electrical Engineering

SWAT Robot

Designed by: Ahmad Hadeed

Supervised by: Prof.Mohamed Zribi

- ## Abstract:

In a hostage situation, one needs to safely monitor the situation without risking the lives of the hostages and the rescuers. The rescue mission will be greatly enhanced by having a camera filming the hostages and sending the information to the rescuers. The SWAT robot is designed for such purposes.

The SWAT robot is a wireless robot controlled remotely by a computer via a joystick. The robot sends live video/audio streams to the rescuers team; it provides valuable information to the rescuers. Moreover, the SWAT robot is equipped with a gun that enables the rescue team to remotely engage and kill the hostage-takers.

The SWAT robot can be easily modified to handle other applications. For example, a modified SWAT robot can be used to detect and to dismantle bombs and/or mines.

Table of Contents

• Introduction:

This project has to do with the design of a SWAT robot, starting from scratch.

After searching for the appropriate design for a graduation project and proposing some designs that has been rejected, a wireless robot unit with a wireless camera that was supposed to be sent for pipe inspection has been suggested by professor M.Zribi.

I then suggested that it would be a SWAT robot that would engage targets if it's required, so it would we a recognizance & attack robot if required.

The design was set to be a wireless robot that is controlled via a joystick, it sends live video feedback & can use an on board weapon to shoot targets.

The project started by trying to interface the joystick to the computer, LabVIEW has been chosen at the end.

Two Boe-bot chassis has been joined together to form the robot's body, the joystick is connected to the computer, the computer analyses the data & sends it serially to a micro-controller that receives the data & sends it wireless through RF unit to the robot. The on broad micro-controller receives the data from the wireless unit & then executes the commands.

• Problem Statement:

It is desired to design a small robot to be used in a hostage situation. In order to fulfill the requirements, the robot has to accomplish the following:

1. The robot should be a wireless standalone unit.
2. The robot should be relatively small in size; enabling it to navigate through tight places.
3. The robot should send a live audio/video stream that enables the SWAT team to have a close look at the situation and enables them to construct their rescue plan.
4. The robot should contain an offense mechanism that enables the controller to engage targets remotely (In this case, a gun).
5. The unit should be controlled via a joystick for the ease of use.

• Design Considerations:

○ *Assumptions:*

The SWAT robot is assumed to operate in a hostage situation, the controller should be able to easily control, maneuver and engage targets when ever he wanted to.

To ease things up, all the controls are available from a single joystick.

Since the robot is able to engage targets by holding a real gun, extra care should be taken to prevent false triggers.

In this prototype unit, it is assumed that the frequency of 433 MHz is not used by any other device nearby and is only dedicated for the robot communication system.

It is also assumed that the frequency of 2.4 GHz channel 4 is also not used by any nearby equipment, as it will be used for the audio/video stream.

The robot is most likely going to operate indoors, the equipment used for communication should have the sufficient power to send signals through walls.

○ *Constrains:*

Time and budget limitations did not allow the usage of powerful motors to drive the robot and enable a faster movement, instead weak and slow servos were used to mobilize the robot.

The effective range of the robot is currently limited by the wireless camera that is currently in use, which has a relatively poor indoor range.

The final mechanical design is not very rugged due to time limitations, final design should look and feel stiffer.

- ***System Environment and External Interfaces:***

  The hardware used in this system consists of:

  1. Laptop + serial port.
  2. DB9 female serial cable.
  3. Joystick.
  4. x2 Boe-bot kits.
  5. RF transceivers.
  6. Camera + Wireless Transmitter.
  7. Wireless Camera Receiver

  The Software used in this system:

  - LabVIEW.
  - Basic Stamp editor. (Inluded with the Boe-bot kits)
  - InterVideo DVR. (Included with the wireless camera)

- ***Budget and Cost:***

  The estimated cost of this prototype system is as follows:

  - Laptop: 350 KD.
  - x2 Boe-bot kits : 100 KD.
  - Joystick: 15 KD.
  - RF transceivers: 35.5 KD.
  - Wireless camera: 48 KD.
  - x5 Servos: 17.5 KD.
  - Estimated total: 564 KD. (Prototype)

- ***Safety:***

  Since the robot is capable of engaging targets remotely, the weapon MUST NOT under any circumstances be false triggered.

  Currently, the design includes a simple communication code that minimizes the possibility of false triggering in case of communication scramble or miscommunication.

- ***Ethics:***

  The robot it's self is a portable compact device that is equipped with a camera that can see in total dark conditions, in wrong hands it might be used to invade privacy.

- ***Performance:***

  The performance of this prototype will not allow it to participate in a real situation, it requires a lot of modifications regarding the speed of mobility, the body structure, weapon handling and other communication issues.

  The final product should have all of listed problems fixed. Note that those problems exists due to hardware limitations due to time and budget limitations.

- ***Documentation:***

  The process of generating the user documents should be rather easy due to the simplicity of usage being in mind while designing the robot.

  The user should not know much technical details to enable him/her to use the robot.

  As for the technical documentation, it would start with the theory of operation and a general overview of what should the robot do and how would it executes it.

  Then the software along with the hardware used should be explained in details.

  Any new updates to either software, hardware or both would require a revision to the technical documentation along with the user documents to insure that they match the new updates.

- ***Design Methodology:***

  The design is made with simplicity in mind, whether it's the simplicity of the design, simplicity of use or maintain. It has also been designed with a relatively low cost in mind.

  This design it's self relies pretty much on existing components. So it would be easier to manufacture and maintain. Broken parts can be easily removed and replaced on site.

- ***Risks and Volatile Areas:***

  It is highly recommended that the surrounding isn't subjected to noise, that is electromagnetic noise, to enable smooth operation.

  The robot hasn't been yet tested with atmospheres containing flammable gases, but the final product should withstand that.

  The current prototype is not designed to withstand humidity or to operate in wet conditions, the final product should handle that well.

  There is no current FCC approval on the RF unit, effects on other electronics and communication devices is not yet known.

- ***Contemporary Engineering Issues:***

  The design uses a powerful software “LabVIEW” that enables easy access and reconfiguration to the robot's control panel. It is fairly said that the whole design is run through “LabVIEW”. This way, virtually any operation can be automated relying on the video feedback form the camera that would be processed on the computer.

  This way, it would make it easier to upgrade and transform into a smart autonomous mode that uses the computer resources to control the robot.

  Nevertheless, it can be programmed to act on it's own in case it loses communications with the controlling computer. (Not implemented at the moment).

• Details of the Design:

- ***Theory of Operation:***

  The SWAT robot is controlled through a joystick that is connected to a computer, the computer will analyze the data and sends it serially to the BS2 (Basic Stamp 2) micro-controller, the micro-controller will receive the serially transmitted data and forwards it serially through the RF wireless unit.

  The micro-controller on the other side (Robot) will be waiting commands to execute. It will receive the RF signal, extracts the information in that signal and sends the appropriate signals to the specified servos, to move/rotate the robot, move the camera/gun, turn on/off the laser and even triggers the weapon.

  A wireless attached camera on board the robot will send a live audio/video stream to the controller.

  The robot it's self can be rotated in all directions. The camera and gun has the ability to move, although it is limited to 180 degrees centered facing forward..

  Once a target is identified and the controller decides to engage it, he/she should switch on the laser, this will locate the position of where the gun is pointed at, the controller will then have to arm the system before he can fire the weapon.

  Different movements of the joystick will decide what direction the robot will start moving to, different buttons on the joystick have different functions such as selecting camera/gun to move, speed of motion of camera/gun, switch laser on/off, reset servos positions and shoot.

The following illustration is a general block diagram of the system.

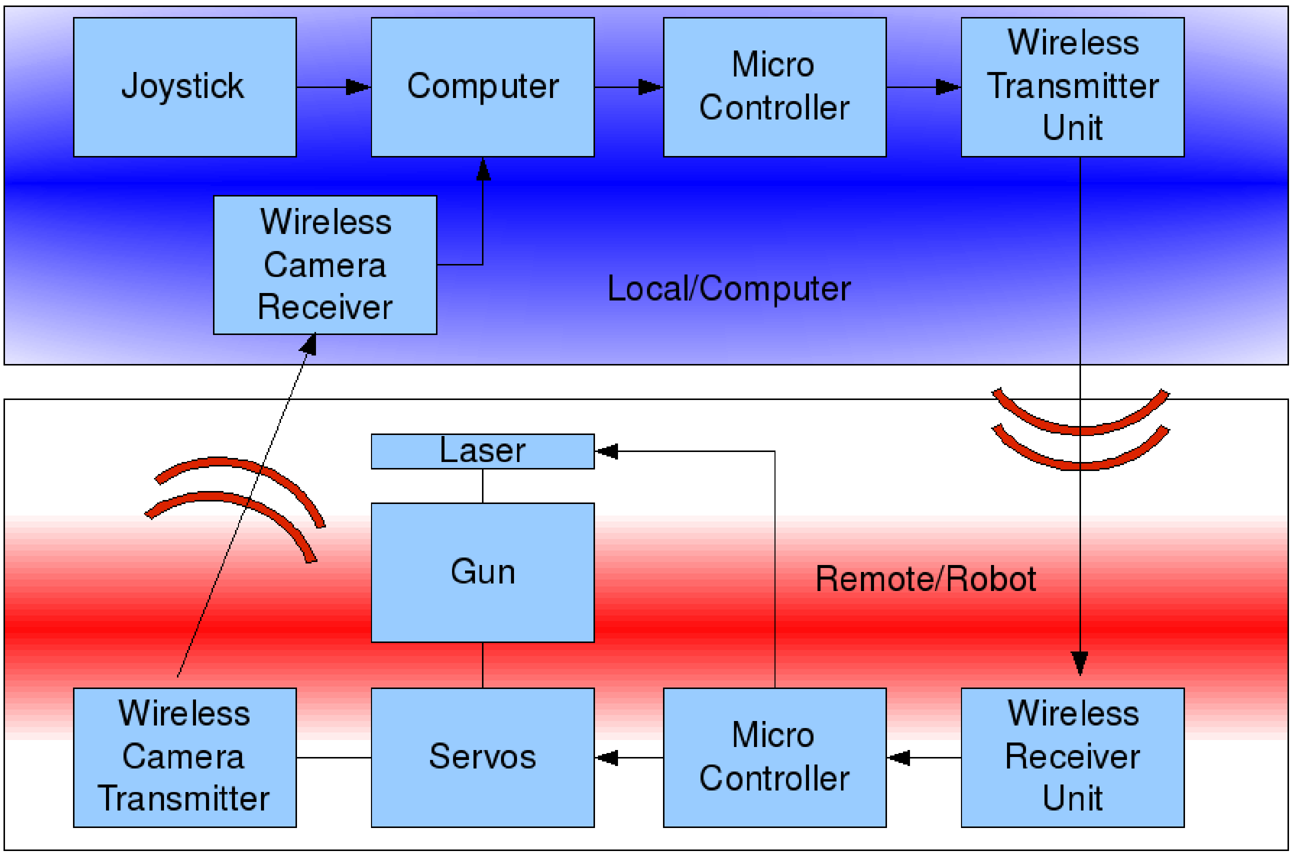


Illustration 1: Block diagram of the full design.

- ***System Specifications:***
  - This SWAT robot is a wireless unit controlled by a remote portable computer.
  - The robot is equipped with e camera that is able to operate in zero light conditions.
  - The final product should contain a fixable weapon stand that can handles different types of weapons.
  - This system uses a laser pointer to indicate the position where the gun is pointed to.

- ***Hardware and Software Requirements:***

  The hardware exists of a joystick, computer with a serial port, a serial port cable with a female adapter at one end and free wires at the other end, BS2 (Basic Stamp 2) micro-controllers (in this case 2 Boe-bot kits), RF transceiver, other servos.

  The software is mainly "LabVIEW" which is being run on a MS Windows based laptop, the robot doesn't need "Basic Stamp Editor" under normal operation. It's only required as a part of the development and upgrading process of the robot.

  - **Hardware Requirements and Design Approach:**

  For the system to perform as required, the data should be transferred successfully from one device to another.

  Starting with the joystick that is connected to the computer via USB (Universal Serial BUS), which should be identified by the software, then monitored for data. The laptop used in this project is equipped with a serial port, which would ease the communication process, so the analyzed data collected from the joystick is sent through the serial port. Then a BS2 micro-controller reads the incoming data from the computer's serial port and forwards it to the other micro-controller on the robot via the RF transceiver. The other micro-controller on the robot will receive the signal, process the data and sends it to the servos and/or other devices such as the laser pointer.

  A wireless camera sends a live audio/video stream which is received by the camera's receiver which is also connected to the computer via USB.

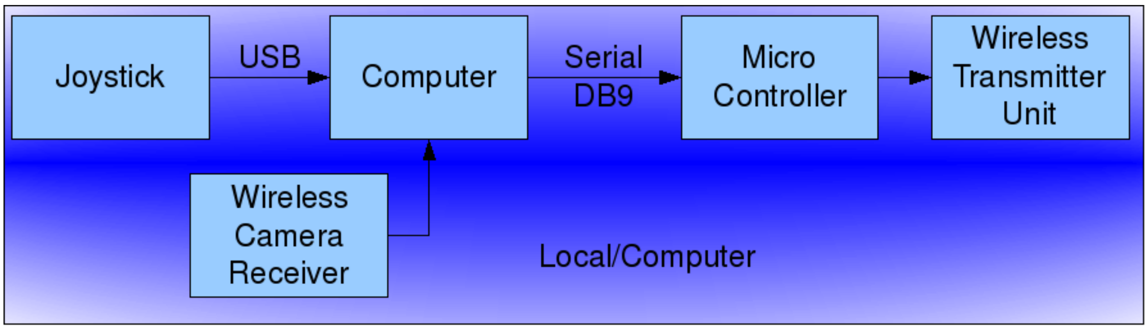


Illustration 2: Block diagram of the local hardware unit.

- **The Joystick:**

  This Logitech joystick has been chosen because of the hat button located on the tip of the stick. In this design, it's used to move the camera/gun.

  The wired version has been chosen over the other wireless version after it has been discovered that the wireless joystick shares the same frequency spectrum of the wireless camera which is 2.4 GHz. This would have an affect on the received audio/video signal after a certain range, were it seems that the joysticks' signal starts masking out the camera signal at the camera's receiver.

Illustration 3: Logitech Extreme 3D Pro Joystick.

- **The Computer:**

  Any portable computer/laptop that has two USB ports (Universal Serial BUS) and a DB9 serial port would do.

  A serial cable connects the computer's serial port with the micro-controller boar.

  At the computer's side, the connector is a 9 pin female D-SUB, while on the other side at the chip, it's separated into it's constructing wires.

  Currently, only 2 pins are required, pin number “5” for common ground and pin number “3” for transmission from computer's side.

- **The serial port:**

  The following figure and table would explain the pins and gives a simple description.

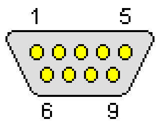


Illustration 4: 9 pin D-SUB male connector at computer side.

DB-9 Pin	Name	Dir	Description
1	CD	←	Carrier Detect
2	RXD	←	Received Data
3	TXD	→	Transmitted Data
4	DTR	→	Data Terminal Ready
5	GND	-	System Ground
6	DSR	←	Data Set Ready
7	RTS	→	Request to Send
8	CTS	←	Clear to Send
9	RI	←	Ring Indicator

- **Basic Stamp 2 Micro-controller:**

  The micro-controller that receives the serial signal and sends it through RF module.

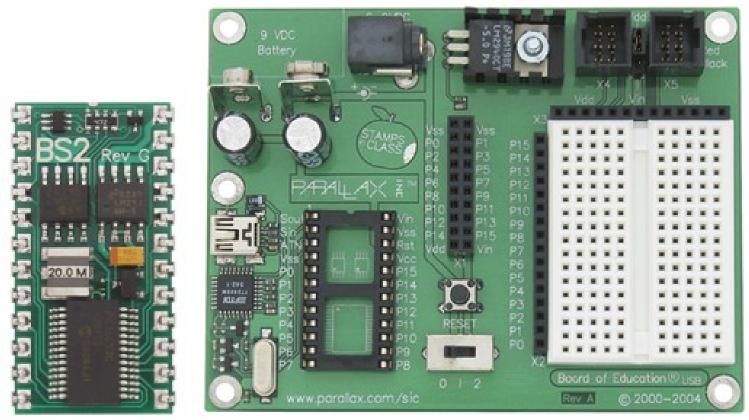


Illustration 5: Basic Stamp 2 micro-controller + board.

This micro-controller comes with the Boe-bot kit. It's relatively easy to program and has got some accessories compatible accessories.

This micro-controller has a duty cycle of 2µs and up to 15 digital I/O ports.

- **RF Module “Wireless Transmitter Unit” (TX/RX):**

  The RF modules are used to transmit and receive signals used to control the robot..

  The modules operates at frequency of 433.92 MHz UHF and can send a signal up to 500+ ft (152 meters), depending on environment conditions.

Illustration 6: RF transmitter module.

Illustration 7: RF receiving module.

- **The 2 joined Boe-bot chases:**

Illustration 8: Original Boe-bot

The above figure shows the original Boe-bot, in this project however, two Boe-bot chases has been joined together to construct a one solid base with 4 wheels drive.

One micro-controller is used on the Boe-bot, as the other is used as described above (send the serial commands).

- **Wireless camera (Transmitter and receiver):**

Illustration 9: Wireless camera + receiver.

The wireless camera is an of shelf item that uses 2.4 GHz frequency channel 4.

It has a built it Infra-red LEDs, which enables it to operate in the dark.

The receiver is connected to the computer via USB, special software “included with the camera” is used to monitor the audio/video stream.

- **Servos:**

There exists 9 servos in the robot.

4 Parallax continuous spinning servos (comes with the Boe-bot kit) used for wheels. Parallax servos has range of (650-850)*2µs duty cycle to operate.

A train of pulses is sent to the servos, 650*2µs is the maximum rotation speed in one direction, 750*2µs is idle and 850*2µs is the maximum rotation speed in the other direction.

Another 5 servos used are Futaba servos, they cannot rotate continuously, instead, they can rotate up to 180 degrees angle.

Futaba servos has a range of (250-1050)*2µs from one maximum end to the other.

Illustration 10: Futaba Servo.

- **Laser pointer:**

  A small hand held laser pointer that is used to locate the position of where the gun is pointed to.

Illustration 11: Hand held laser pointer.

The switch of the laser module is kept closed, the voltage to power the unit is sent via the micro-controller when the laser module should be turned on.

- **Weapon:**

  In this prototype, an electric semi-auto BB gun is used for demonstration.

Software Requirements and Design Approach:

LabVIEW has been the chosen software due to it's ease of use and ability to to communicate with different attached hardware on the computer.

The design approach is simple, on the computer, initiate joystick, get data, reformat data and send them to the computer's serial port.

The 1st micro-controller, just receives the serial data from the computer and forwards it to the RF unit.

The robot should then receive the wireless data and executes it.

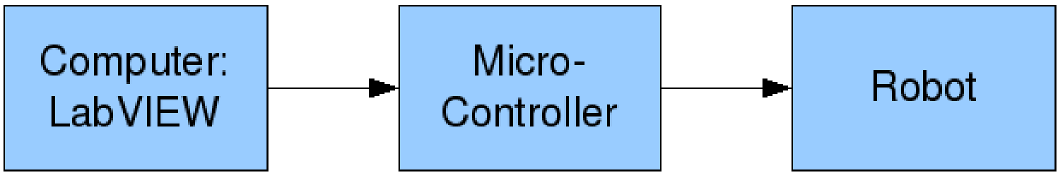


Illustration 12: General Software Design.

- **LabVIEW (Computer):**

  The program is separated into modules. The 1st module is called the modular design, which includes other modules and functions.

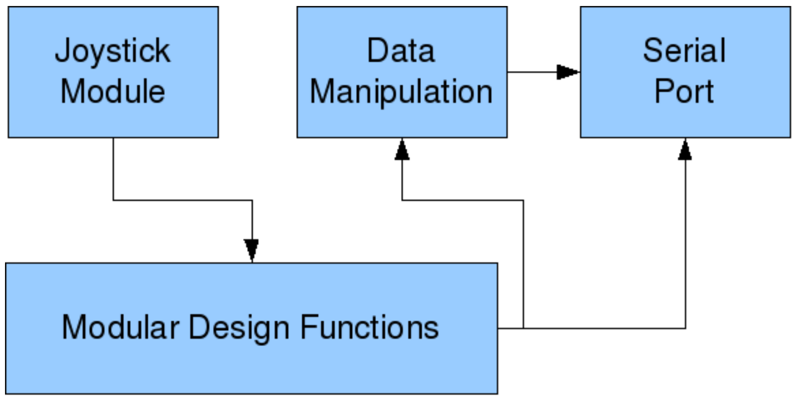


Illustration 13: Modular module general overview

  The modular design, contains the joystick module, data manipulation module, serial port module and other internal functionality, which will be discussed after the joysticks' module..

- **Joystick module:**

  The joystick module is specialized in identifying and initialize the hardware "joystick", get variables from different, axises or buttons. The joystick module was the 1st module to be designed.

  In the early stages of the design, it was the only module and had extra functionality such as manipulating some joystick data and creating two variables, which are used as speeds for the right and left servo wheels. The module still exports those two values to be used directly by the serial module.

  The joysticks' axises gives values ranging from (-32768~32768), such resolution is not necessary. We've discussed that our servos that are used for wheels uses a range of
  (650-850)*2µs, so the needed range is, 850-650=200. Instead of sending the actual values of (650-850), sending data in the range of 0 to 200 will reduce bandwidth and original data can be restored on the other side.
  For example: The idle state requires 750, so we would send 100 and then add a value of 650 for each on the robot's side. If we send 0, 0 + 650 = 650.
  The joystick gives a position of it's 3 axises, x,y and z. The x and y will determine the speed and direction of which the robot will be moving with. The direct x and y values cannot be used as raw numbers and should be transformed to suit four conditions representing the four quadrants of the circle.
  This means the x and y joystick numbers should be transformed to the right and left speeds for the servos. This is done by writing a special code for each of the quadrants of the joystick, this will use the x and y values of the joystick to generate the appropriate signal for the wheel servos. The z axis is used to rotate the robot in it's position around it's z axis (vertical).
  Other buttons are extracted and connected to ports in the module for further usage.
  Note that in the previous process, 2 values ranging from (0-200) where generated and can be transmitted later using two bytes. 1 byte for each wheel servo, right and left.
  Since the system is a 4 wheel drive, 1 byte will be sent to 2 servos.

In general, the joystick module will do the following:

- Identify and initiate the hardware.
- Decrease the axises resolution for a suitable (0-200) range.
- Use the x, y and z axis's values to generate a suitable signal for the wheel servos.
- Assign other joystick buttons as an external ports to be used later.
- Continuously monitor the joystick with a sampling time of 5 ms.

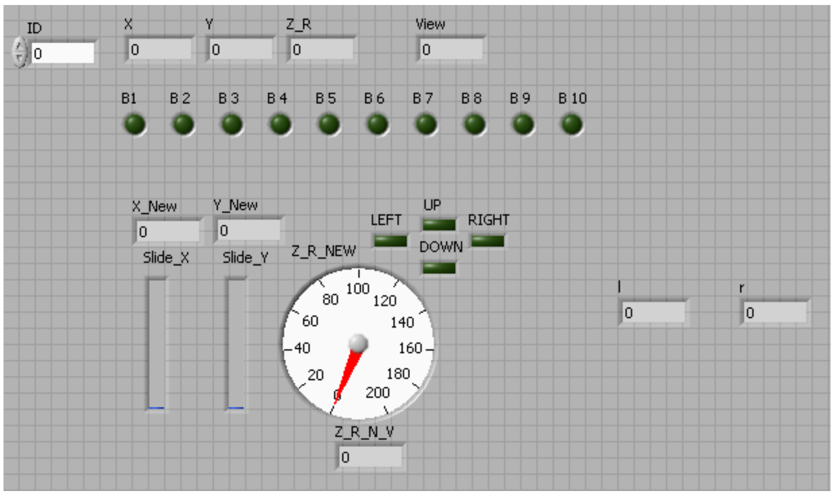


Illustration 14: Joysticks' module front panel

Table 01: Listing the joystick buttons and their functionality.	
Button	Function
1	Shoot
2	Laser on/off
3	Decrease movement speed
4	Increase movement speed
5	Select camera
6	Select gun
9	Reset servo positions
10	Arm system on/off
Hat	Move camera/gun

Note: those are the assumed functions that should be implemented in the next stages of the design.

- Brief description of the functionality:
  - Shoot: To trigger the weapon.
  - Laser on/off: toggle laser on/off.
  - Decrease movement speed: Decrease the speed of the servos holding the camera/gun.
  - Increase movement speed: Increase the speed of the servos holding the camera/gun.
  - Select camera: Will select the camera servos to move.
  - Select Gun: Will select the gun servos to move.
  - Reset servo positions: Will send a signal to move all the servos holding the camera and gun to their center position.
  - Arm the system: Enables the shoot function. So even if you press the shoot button, the system will not shoot unless it is in the armed state. Toggles on/off.
  - Hat: The hat is the small movable directional switch on the top of the joystick, this will move the selected camera/gun to move in the direction we specify, with the speed we've set.

- **The modular module:**
  The modular module (represented in illustration 13), is the module that contains all other modules in the design, it also has it's own functionalities and provides the final user's front panel.
  From the joystick module, we have got 2 values for the wheel servos plus other buttons.
  The 2 speed values are sent to the serial module.
  The buttons will be programmed now to perform the assigned tasks.
  - Laser button:
    A special function is written to enables toggling of the laser when the button is pressed, a button hold technique is used here. For example if you keep the button pressed, the laser will toggle on/off continuously, the hold function is used to detect that the button is released before toggling again.
    The laser signal is sent to the data manipulation module.
  - Armed system button:
    Has the same technique used for the laser as for toggling and hold function, but this one generates a system armed signal.
  - Shoot button:
    The shoot button is logically anded with the armed signal and then sent to the data manipulation module.
  - Increase/Decrease movement speed:
    A function is written for the associated buttons so that a value between 1 and 100 is generated to control the speed of the servos moving the camera/gun, with button 3 decreasing the value and 4 increasing the value without going of limits.
    This value is sent to the serial module.
  - Select Camera/Gun:
    Buttons 5 and 6 are sent to a function that toggles between camera/gun, the function sends one bit, '0' if the camera was chosen and '1' if the gun was chosen.
    This bit is send to the data manipulation module.
  - Reset servo positions:
    One bit that is sent to the data manipulation module.
  - Hat:
    The hat provides as a switch with 8 directions with up, down, right and left directions.

The function will take the values and provide x=-1,0,1 and y=-1,0,1.
The new x and y values are send to the data manipulation module.

In general the modular module will do the following:

- Sends the 2 speed values from the joystick (used for the wheels) to the serial module.
- Toggles laser signal and sends it to data manipulation module.
- Toggles system arm signal.
- Logically ANDs the shoot and arm signal and sends it to the data manipulation module.
- Generates a speed variable to be used with servos controlling camera/gun and sends it to the data manipulation module.
- Toggles one bit to select either camera/gun and sends it to the data manipulation module.
- Sends the joystick button for the servo reset to the data manipulation module.

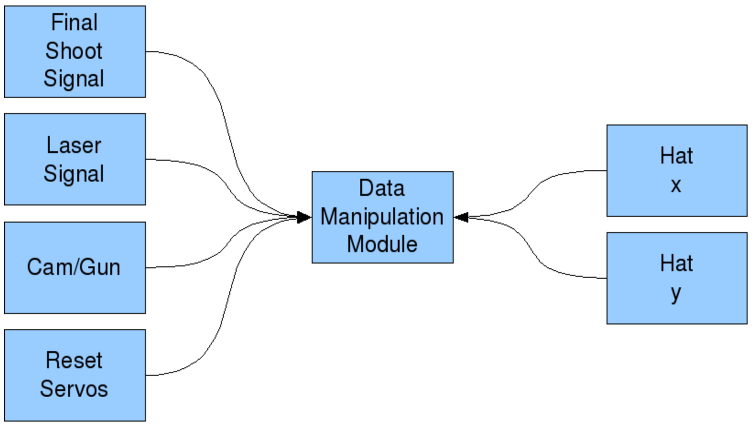


Illustration 15: Modular module and the Data manipulation module.

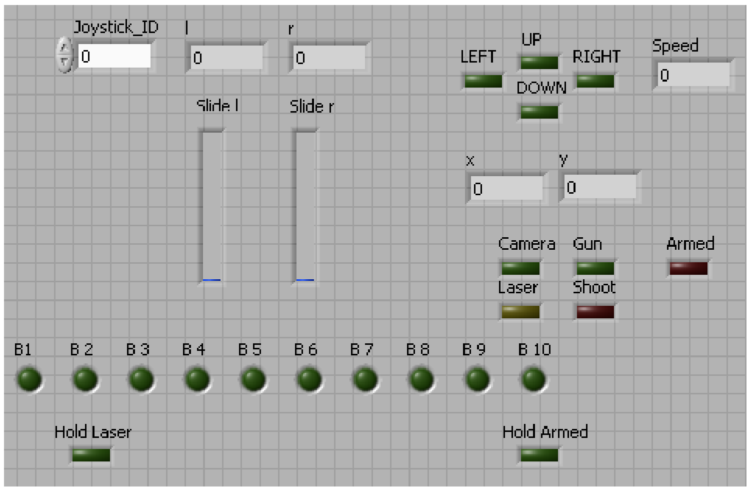


Illustration 16: Modular module front panel.

- **Data manipulation module:**
  The data manipulation module is to collect the bits and other values to construct one byte that contains all the data to be sent to the robot.

  The x and y directions are sent to a function that eliminates the negative sign for further usage in serial communications, as we are not going to use signed integers to represent the binary values. The function will generate xo, xxo, yo and yyo bits.
  For a negative x direction xo=0,positive x direction, xo=1, the xxo=0 when there is no x direction movement and xxo=1 when there is a movement, either -x or +x. The same applies for yo and yyo.
  The previous 4 bits along with the reset, cam/gun and laser and shoot signals, total of 7 bits are sent to the bits_byte_string module, that will combine the bits into a byte.
  The shoot signal will be sent as a full byte for safety reasons.
  It has been realized that in rare conditions, some communication errors might happen during data transfer due to the environment, the gun would trigger but it's self when it's bit signal is switched to “1” during communications. So it has been decided to send the a value of decimal “85” or binary “01010101” byte to minimize the chance of accidental weapon trigger.
  Other communication errors which happens rarely won't be considered as a serious issue.

- **Bit_Byte_string:**
  This module will arrange the bits in a certain order to construct the control byte that will be sent to the micro-controller later.
  The byte is constructed by multiplying each bit by the location it occupies in the byte and summing them up.
  The following row represents the final byte.

0	xo	xxo	yo	yyo	Cam/Gun	Laser	Reset

  To construct that bit:
  Reset=1*2^0
  Laser=1*2^1
  Cam/Gun=1*2^2
  yyo=1*2^3
  yo=1*2^4
  xxo=1*2^5
  x=1*2^6

  Then the results will be summed up and would represent the final byte.
  The final byte will be sent through the data manipulation module to the serial module.

- **Serial Module:**
  The serial module is the module responsible for computer's serial communication, 1st it will detect and initiate the hardware and then it will receive the bytes needed to be sent and sends them serially at the specified protocols and baud rate.

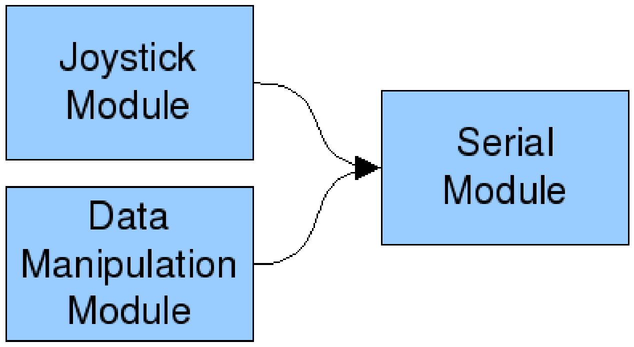


Illustration 17: Serial module relation with other modules.

In all our previous numbers and bytes, we did not exceed the decimal value “200”. The last highest order bit the the control bit was left '0' on purpose, the weapon's trigger highest bit was also left '0' , this is because we want to use other values larger than “200” for control issues, in this case, the serial port uses the decimal value of “201” for start receive signal to the micro-controller.
First, it sends the decimal “201” to tell the micro-controller to start to receive.
It then sends the bytes one by one.
The right servo speeds will be transformed into a byte string and sent, then the left servo speeds will be transformed into a byte string and sent, then the speed of camera/gun servos will be transformed and sent, then we have two ready bytes, the control byte containing hat direction, cam/gun, laser and reset. At last but not least the shoot byte.

The following are the serial communication parameters:

Baud Rate	9600 b/s
Number of bits	8
Parity	None
Flow Control	None

In general, the serial module will do the following:

- Identifies and initiates hardware and communication parameters.
- Receives data to be transmitted.
- Sends a start receive signal.
- Converts data to byte if needed.
- Sends a sequence of bytes serially.

- Micro-controller #1:
  This micro controller acts as a data redirector, where is receives the data being sent from the computer's serial port & sends it to the RF wireless module.
  It just simply waits for start receive signal, then it receives all the data to be transmitted, then sends them to the wireless transmission unit.

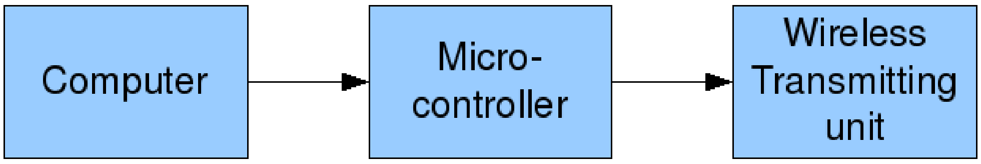


Illustration 18: Micro-controller #1 operation.

- Micro-controller #2 (Robot):
  The 2nd micro-controller is the brain of the robot, it starts by resetting all servos to their middle values/positions. It then waits for start receive signal, starts receiving the data from the wireless unit, identifies different parameters, reformat for use & execute.
  Upon the execution of the code, signals are sent to the servos to locate their new positions & laser to turn it on/off.

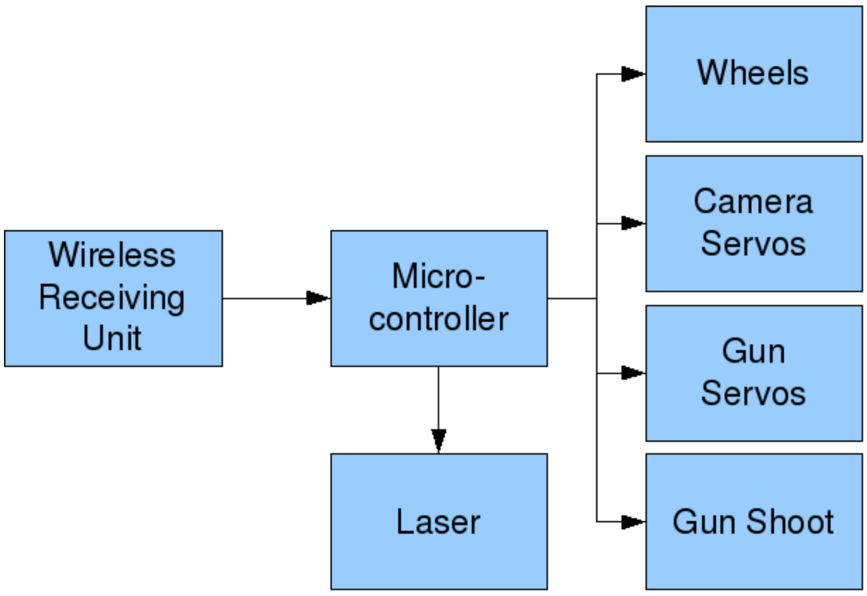


Illustration 19: Micro-controller #2 operation.

- ***Test and Demonstration:***

  - **Test:**

    The design has been has been tested module by module at the beginning, then after assembling the modules together the whole design has been tested and performed as expected.

    Some small glitches has been discovered through testing and has been sorrected successfully.

  - **Demonstration:**

    The demonstration would be a simple run of the SWAT robot & showing it's capabilities.

    The demonstration would also include a brief view on the LabVIEW design modules to clarify the image.

- ***Financial Information:***

Item	USD	KWD	Date
Joystick	-	15	06/01/09
Parallax RF units	129.55	35.68	27/10/08
Delivery Charges	-	3	11/11/08
Wireless Camera	169.44	47.66	21/11/08
Parallax Extra Cables & IR unit	31.09	8.65	17/11/08
Delivery Charges	-	8	01/12/08
Total	**-**	**117.99**	

A laptop omputer would cost another 300 KD (Software is currently excluded from the cost estimate).
Laser pointer about 3 KD. (already existing item)
A semi auto BB-gun arround 10 KD. (already existing item)
Two Boe-bot kits has been provided by professor, plus another few servos.

Results and Discussion:

Results of the Project:

A SWAT robot has been designed and implemented, it has been presented & demonstrated to the professors of the department

It performs all tasks given & satisfies all requirements of the design.

Discussion of the Results of the project:

The outcome was a wireless robot that can be controlled via a joystick connected to a computer. It's able to send live video/audio streams to the computer. The camera & gun can be rotated & moved according to needs.

A laser that is attached would be activated to indicate the position or place that the gun would shoot. Then it can engage a target by firing the weapon.

Lessons Learned:

- When working with a project, thinking of multi stages in parallel is required to avoid future incompatibilities.
- Working alone with a huge project might be good in terms of having all thoughts organized by a single person, but it's always good to have a helping hand from someone trusted, specially of you are time limited.

Team's Performance:

Not applicable, the job has been done by one person.

Conclusion and Future work:

Conclusions:

The project was a great success, it achieved all requirements & goals.

Future Work:

- Send the robot to dangerous & life threatening environments.
- Equip robot with different sensors to be able to detect hazardous or radioactive materials in a certain environment.
- Equip with extra robotic arm to enable mine or bomb disassembly.
- Send robot into tight places such as vents or pipes for discovery purposes.
- Enable autonomous mode through image processing.

- **References:**

- http://www.computerhope.com/help/serial.htm
- Parallax Boe-bot user manual.
- Parallax RF unit manual and datasheet.

- ## Appendix A:

Ahmad Hadeed

Kuwait

telephone: +965-97968044

e-mail: hadeed@hadeed-power.com

website: www.hadeed-power.com

Education:

Kuwait university

College of Engineering,, Department of Electrical Engineering. (2008/2009)

Advisor: Prof. Mohamed Zribi

- ## Appendix B:
  - ### ***Parts List:***
    - x2 Boe-bot kits.
    - Parallax Transceiver set.
    - Laser pointer module.
    - x5 servos.
    - Extra cables/pins.
    - Serial cable.
    - Weapon.

- ## Appendix C:

  - ### *Equipment List:*

    - Laptop Computer.
    - Joystick.
    - Wireless camera.

- ## Appendix D:

  - ### *Software List:*

    - LabVIEW.
    - BASIC Stamp Eitor (Included with Boe-bot kits).

- **Appendix E:**
    - ***Special Resources:***

- **Appendix F:**
  - ***User's Manual:***
    - **Software Installation:**
      - Install LabVIEW.
      - Install InterVideo WinDVR (comes with the wireless camera)
    - **Connections:**
      - Connect the Joystick to the USB port of the laptop.
      - Connect the Camera receiver to the other USB port of the laptop.
      - Connect the serial port of the computer to the micro-controller #1.
      - Connect the batteries of both micro-controllers.
      - Connect the camera with the 9V battery.
    - **Software Initialization:**
      - Start LabVIEW by opening the modular design front panel 01_Module.vi.
      - Run the program.
      - Select the proper joystick ID (Default 0, if it was the 1st USB device that was connected to the computer).
      - Run InterVideo WinDVR.
      - Choose correct camera channel (Ch 4 used in this project).
    - **Usage:**
      - **Robot Movements:**

The joystick acts as a controller for the robot, each axis movements will move the robot in a different direction. It's a simple combination that can be realized after little experimenting with it.

The y axis will give the forward or reverse speed, while the combination of x and y axises will start a differential speeds between right and left wheels that causes rotation of the robot.

The z axis of the joystick will move the robot around it's self, note that the z axis function will not work if the joystick has values in x or y directions. Meaning it should be centered, otherwise the x & y directions will be taken instead of the z axis, because they have a higher priority. (You cannot move the robot around it's self while it's moving in the 1st place!)

- **Choosing Camera/Gun:**

  To select the camera press button 3 on the joystick, to select the gun press button 4 on the joystick.
- **Choose the speed of the camera/gun movement:**

  The buttons 5 and 6 decreases & increases the speed respectively
- **Move the Camera/Gun:**

  To start moving the camera/gun (whatever is selected), you press the view switch in the direction desired.
- **Laser ON/OFF:**

  Press button 2 on the joystick to toggle the laser on/off. (The laser will help you locate the position where the gun is pointing at)
- **Arming the System:**

  Before shooting, the system should be armed, press the button number 10 on the joystick to arm the system.
- **Shooting the Weapon:**

  After the system has been armed, press button number 1 on the joystick to engage the target.
- **Reset the Servos:**

  In case the user has lost the sense of directions, lost the camera or gun positions, he can simply press the joysticks' button number 9. This will reset the servos positions holding the camera and gun.

Appendix G:

○ *LabVIEW Programming:*

▪ The Joystick Module:

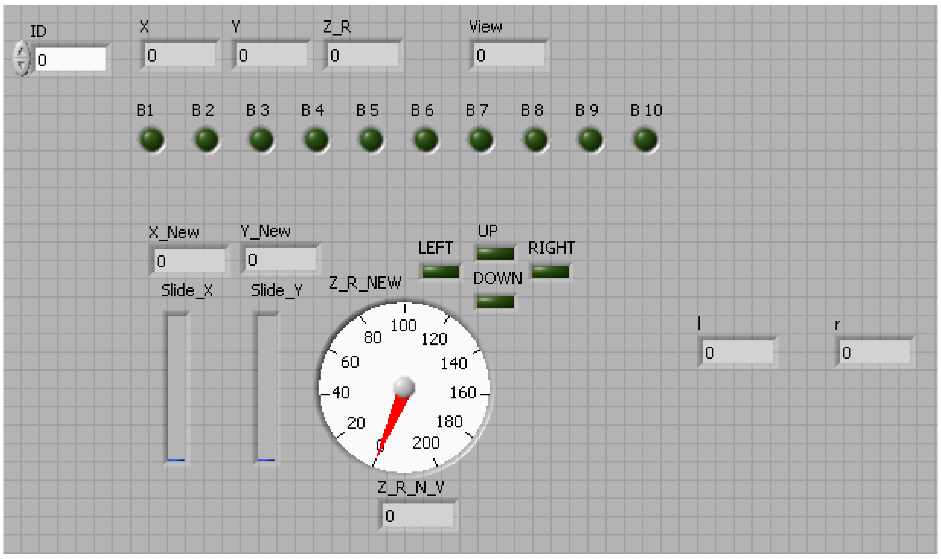


Illustration 20: Joystick module front panel.

Running the joysticks' module can let you monitor all the axises values & buttons of the joystick.

This panel has:

- ID: Is the joysticks' COM configuration (Default 0, if Joystick was 1st USB device to be connected to the computer).
- x: Representing the x-axis of the joystick.
- y: Representing the y-axis of the joystick.
- Z_R: Represents the z-axis rotation.
- B1-B10: The joysticks' buttons.
- X_New: The new reduced x-axis values.
- Y_New: The new reduced y-axis values.
- Z_R_NEW and Z_R_N_V: Represents the new reduced vales & location of the z-axis.
- UP,RIGHT,LEFT and DOWN: Gives the position of the view switch.
- l and r: Represents the final values that is sent to the wheel servos (right and left servos).

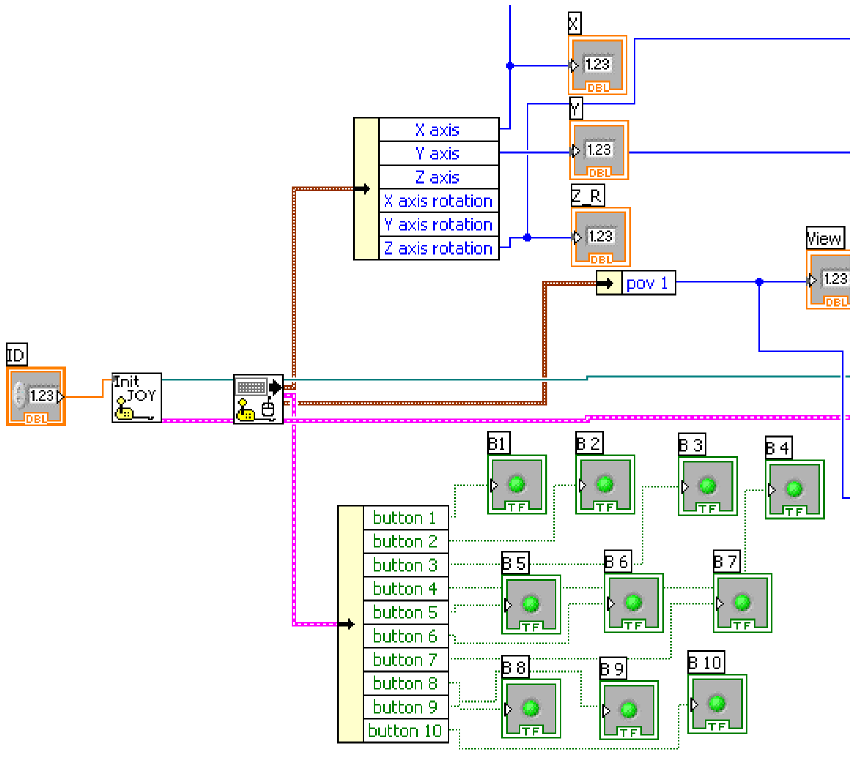
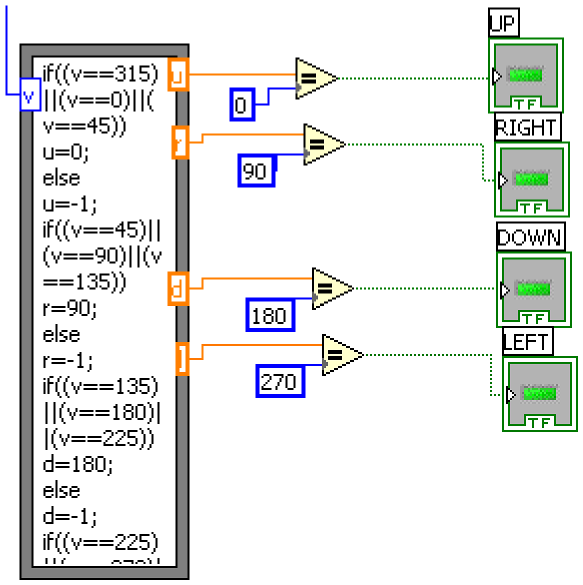


Illustration 21: Joysticks' block diagram showing initialization & parameter extraction.

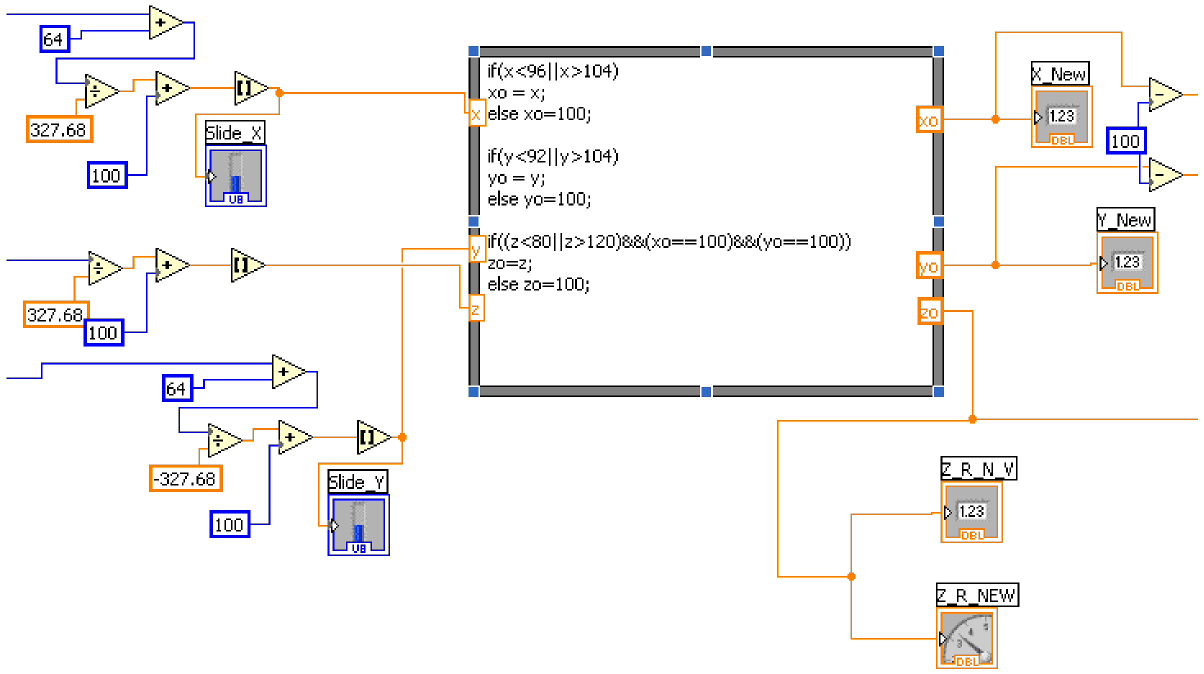
The above figure shows the joystick module in LabVIEW with all axises & button parameters being extracted.



```
if((v==315)
||(v==0)||(
v==45))
u=0;
else
u=-1;
if((v==45)||
(v==90)||(v
==135))
r=90;
else
r=-1;
if((v==135)
||(v==180)|
|(v==225))
d=180;
else
d=-1;
if((v==225)
```

Illustration 22: Joysticks' block diagram showing the view switch signal converted to UP, RIGHT, DOWN and LEFT.

In illustration 22, the “v” signal from the view switch which has either 0, 45, 90, 135, 180, 225, 270 and 315 values is transformed into boolean values indicating UP, RIGHT, DOWN and LEFT positions of the switch.



```
if(x<96||x>104)
xo = x;
else xo=100;

if(y<92||y>104)
yo = y;
else yo=100;

if((z<80||z>120)&&(xo==100)&&(yo==100))
zo=z;
else zo=100;
```

Illustration 23: Joysticks' block diagram showing 1^{st} *signal conditioning for the joysticks' axises parameters.*

The resolution of the joysticks' axises ranges from (-32768~ 32768) and the servos only needs a range of 200 states to fully function.

Note that we have added the value of “64” because the joystick has an initial value of “-64” that is shifted from the center.

We do signal conditioning, we divide by “327.68”, if the value has been initially “0” then the output of the division is “0”, for maximum values (-32768/ 327.68)=-100 and (32768/ 327.68)=100.

Then the value of “100” is added, so that the “-100” becomes “0” & the value of “100” becomes 200.

So basically now, we have a range of (0~200) which is 200.

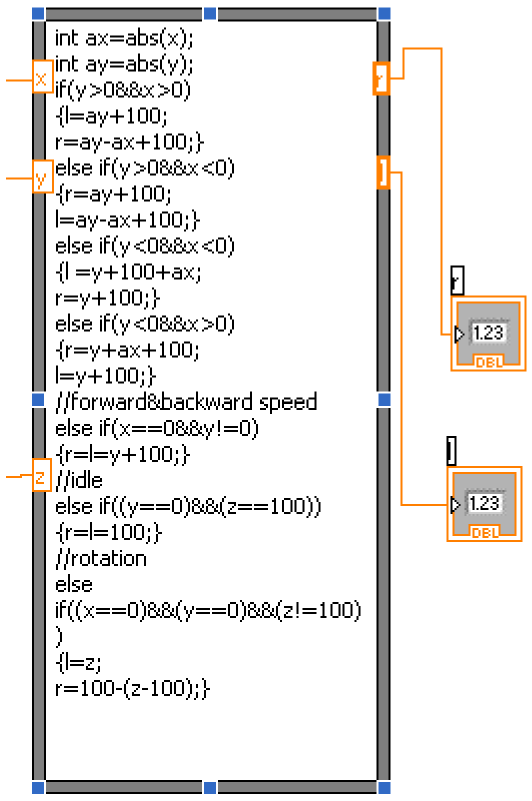
Note: The “y” axis is divided by a “negative” number, that is because when using joysticks for flight simulators, the axis is inverted (Pulling down the joystick gives a positive number).

Illustration 24 shows the formula node that transforms the x & y directions of the joystick to the r & l “right & left” speeds for the servos. It basically consists of the x & y directions for the 4 quadrants of the circle, idle state & forward/backward movements.

If the joystick is moving in the y direction, then both r & l values have the same value of y (r=l=y), when x value is represented, r & l values change depending on the quadrant.

For example, if the joystick is having y values only, both r & l have the same value, if x started to have a positive value the r starts decreasing as x increases. This will give a speed difference between right & left that shifts the robot to the right. If the x starts to have a negative direction, then the the value is decreased from l & the robot starts to rotate left...etc.

This applies to the other two quadrants but in the opposite way, just the same way you drive your car & rotate your steering wheel.



```
int ax=abs(x);
int ay=abs(y);
if(y>0&&x>0)
{l=ay+100;
r=ay-ax+100;}
else if(y>0&&x<0)
{r=ay+100;
l=ay-ax+100;}
else if(y<0&&x<0)
{l =y+100+ax;
r=y+100;}
else if(y<0&&x>0)
{r=y+ax+100;
l=y+100;}
//forward&backward speed
else if(x==0&&y!=0)
{r=l=y+100;}
//idle
else if((y==0)&&(z==100))
{r=l=100;}
//rotation
else
if((x==0)&&(y==0)&&(z!=100)
)
{l=z;
r=100-(z-100);}
```

Illustration 24: Joysticks' block diagram showing the adjustment of r and l for the four quadrants.

When the z-axis of the joystick is rotated (without moving in x & y direction), the r and l values gets shifted of center equally to rotate the robot around it's z-axis

▪ The Modular Design:

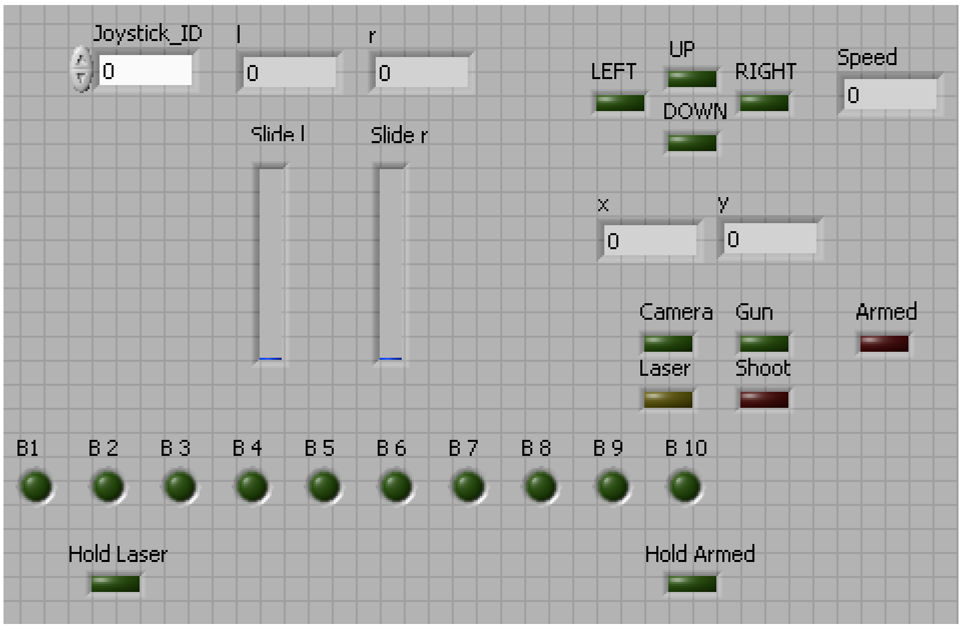
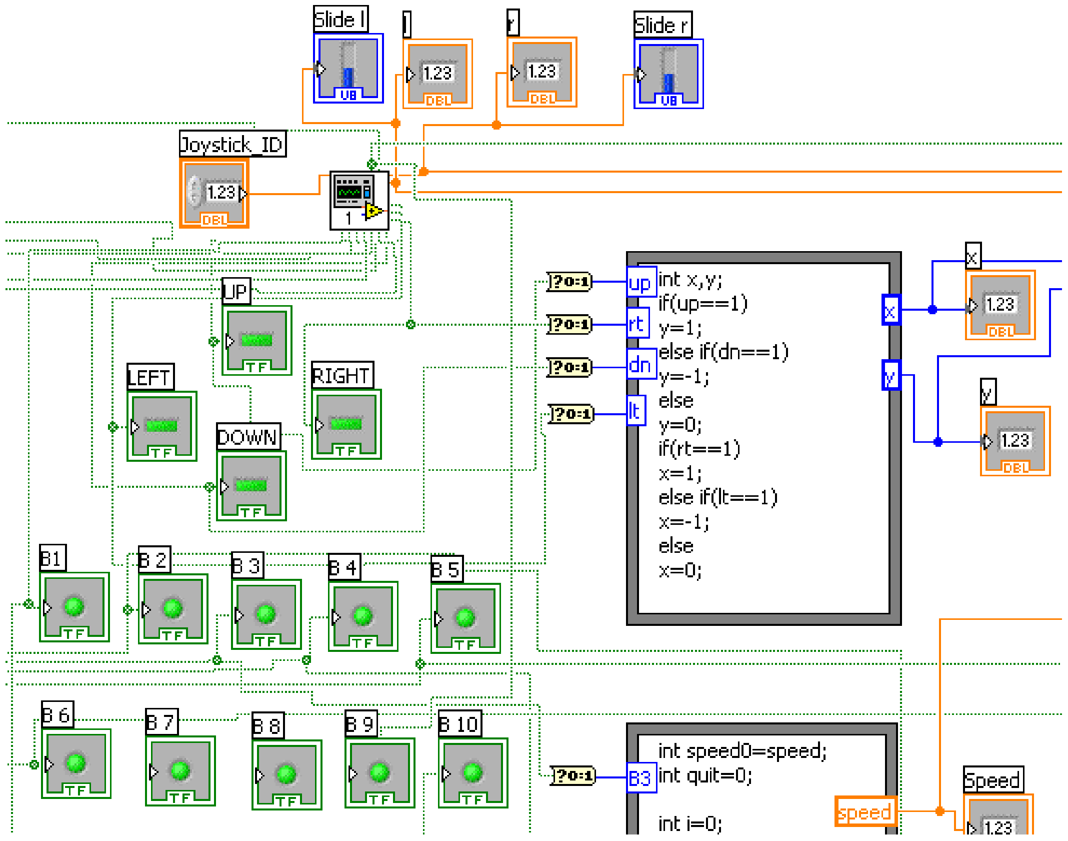


Illustration 25: Modular design front panel.

This is the front panel that the user will see when working with the robot, this panel indicates the commands sent to the robot.

This panel had:

- Joystick_ID: Is the joysticks' COM configuration (Default 0, if Joystick was 1st USB device to be connected to the computer).
- l and r: Representing the speed of the left & right wheel servos.
- Slide l and r : Graphical representation of l and r values.
- UP,RIGHT,LEFT and DOWN: Gives the position of the view switch.
- x: The x value of the view switch.
- y: The y value of the view switch.
- Speed: The speed of movement for the servos holding the camera and gun.
- B1~B10: Joystick buttons indication.
- Hold Laser, Hold Armed: Button hold indication (Not important for user, only for debugging purposes).
- Camera: Indicating that the camera is chosen to be moved.
- Gun: indicating that the gun is chosen to be moved.
- Armed: Indicating the system status, safe or armed & ready to shoot.
- Laser: Indicates if the laser is switched on or off.
- Shoot: Will illuminate if the system is armed & button 1 is pressed.



```
int x,y;
if(up==1)
y=1;
else if(dn==1)
y=-1;
else
y=0;
if(rt==1)
x=1;
else if(lt==1)
x=-1;
else
x=0;
```

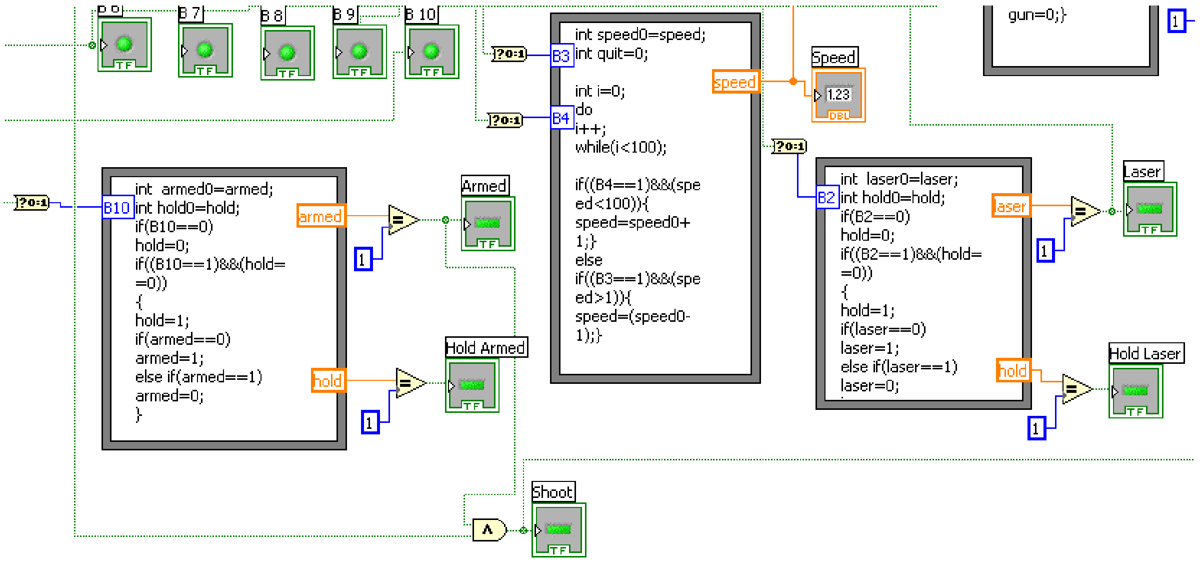
```
int speed0=speed;
int quit=0;

int i=0;
```

Illustration 26: Modular design block diagram showing joystick module, other parameters and buttons and preview switch 1st signal conditioning.

The joystick module is presented as a small block at the left upper side of the image, buttons and other parameters has been extracted to show the status on the front panel screen.

The buttons (UP, RIGHT, LEFT, DOWN) has been transformed from boolean for each to x and y negative and positive.



```
int armed0=armed;
int hold0=hold;
if(B10==0)
hold=0;
if((B10==1)&&(hold=
=0))
{
hold=1;
if(armed==0)
armed=1;
else if(armed==1)
armed=0;
}
```

```
int speed0=speed;
int quit=0;

int i=0;
do
i++;
while(i<100);

if((B4==1)&&(spe
ed<100)){
speed=speed0+
1;}
else
if((B3==1)&&(spe
ed>1)){
speed=(speed0-
1);}
```

```
gun=0;}
```

```
int laser0=laser;
int hold0=hold;
if(B2==0)
hold=0;
if((B2==1)&&(hold=
=0))
{
hold=1;
if(laser==0)
laser=1;
else if(laser==1)
laser=0;
```

Illustration 27: Modular design block diagram showing arm, speed & laser.

Illustration 27 shows extra functions such as arm mode, speed for the servos holding camera/gun & laser on/off function.

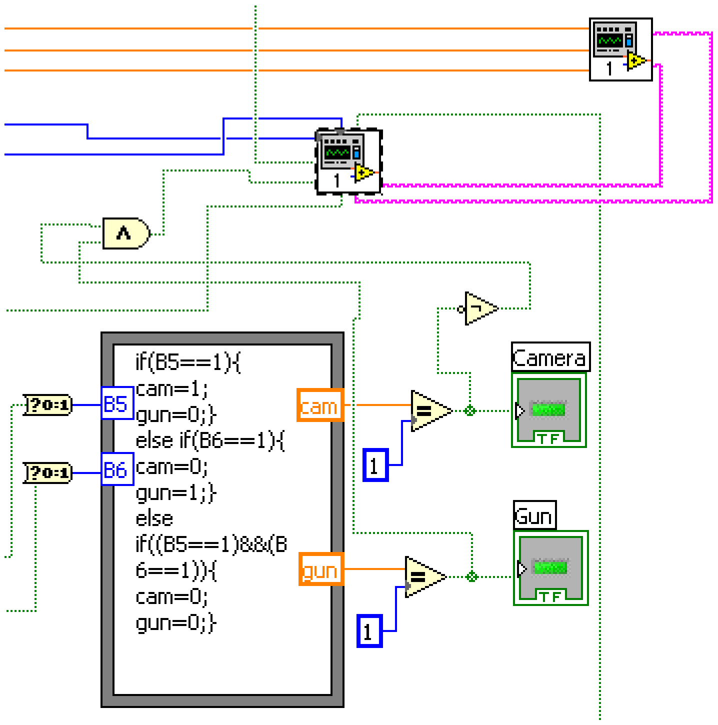
In the arm and laser functions, a hold technique is used to prevent continuous switching while the button on the joystick is kept pressed.

When the button for the arm function or laser on/off is pressed, the formula node detects that the button has been pressed and then checks the previous status of the system & performs the opposite operation, toggling between on/off. It then sends the boolean value to LEDs on the front panel to indicate the status.

The problem with LabVIEW is that it resets the variables on each cycle of the program, that will reset the status every time, to fix this problem, the output value is used as an input value in the same formula node.

A hold function will keep the code from leaving a loop until the button is released, so it won't toggle the status tens of times a second.

The servos speed controlling camera/gun is controlled by a function that sets the speed of movement from 1 to 99. The value starts with 0 in LabVIEW, you should then use the joystick buttons that controls the speed to increase or decrease the speed. Again, the output variable is used as an input variable again to prevent the reset of the value on each cycle. The speed value is sent to an indicator "Speed" on the front panel to show the current speed.



```
if(B5==1){
cam=1;
gun=0;}
else if(B6==1){
cam=0;
gun=1;}
else
if((B5==1)&&(B
6==1)){
cam=0;
gun=0;}
```

Illustration 28: Modular design block diagram showing camera/gun selection, data manipulation & serial modules.

Illustration 28 shows a part of the modular design where either the camera or gun is selected to be moved. The formula node detects which button on the joystick has been pressed & switches between camera & gun. It then sends the boolean value to LED indicators located on the front panel to show which is selected.

The boolean value of the camera is inverted and logically anded with the gun's boolean value, this will give a boolean value of "0" when the camera is selected and a boolean value of "1" when the gun is selected.

All the previous parameters, camera/gun selection, preview switch position, armed status laser, reset and shoot are sent to the data manipulation module located in the center top of illustration 28. The speed parameter is sent directly to the serial module located on the right top of illustration 28.

The output of the data manipulation “2 bytes” are sent to the serial module.

▪ Data Manipulation Module:

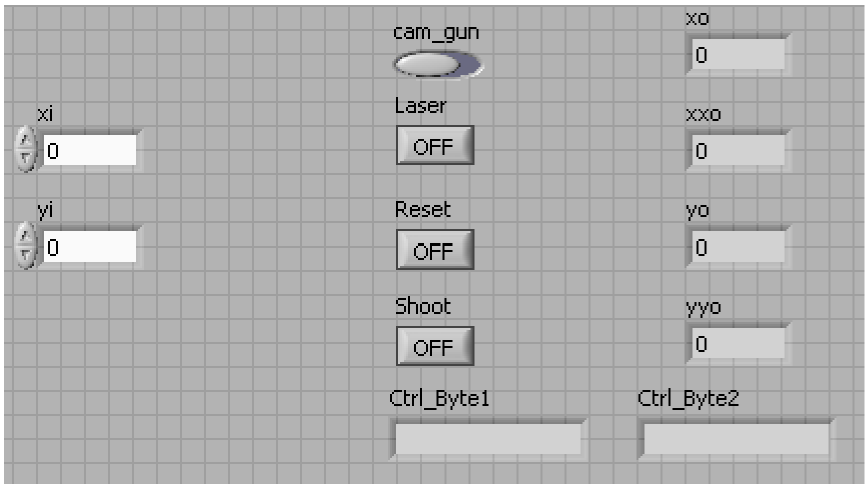
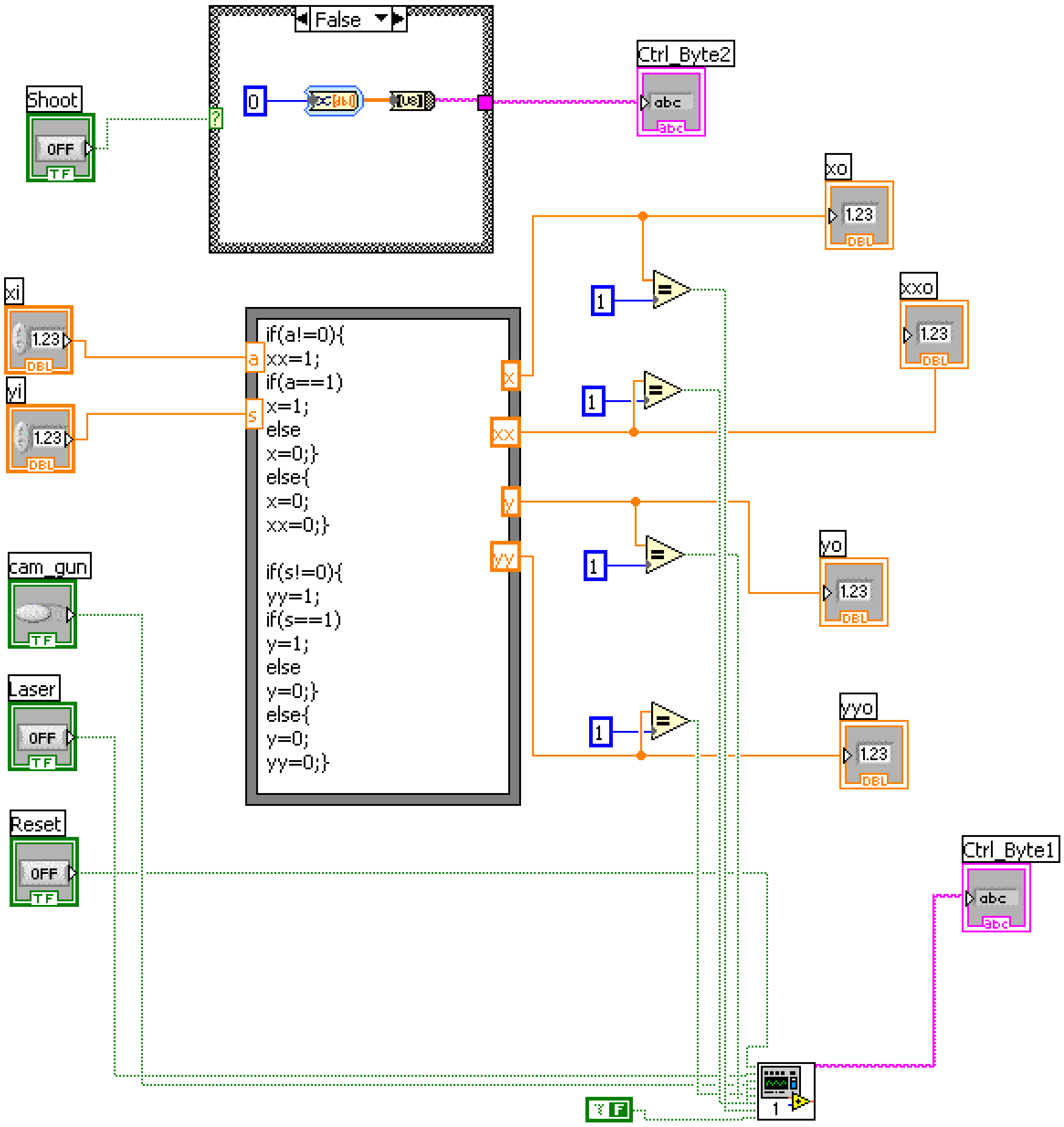


Illustration 29: Data manipulation front panel.

The data manipulation module does the job of converting the different parameters received from the joystick into a usable data that can be sent easily through serial transmission.

The data manipulation module receives the the x, y view switch positions & converts them into 4 bits.

- xi, yi: Input that represents the x and y values of the view switch.
- cam_gun: Input that represents the chosen equipment to be controlled “0” for camera and “1” for the gun.
- Laser: Input represents the status of the laser.
- Reset: Input represents the reset signals for the servos.
- Shoot: Input that represents the shoot signal.
- xo, yo: Outputs that represents the x and y directions of the view switch, “0” indicates left while “1” indicates right.
- xxo, yyo: Outputs that represents the status of the view switch, idle or active. (Note we have xo and yo to be either left or right, we need another indicator for idle or active). The value of “0” indicates idle and the value of xo and yo are irrelevant. If the value is “1” then the value of xo and yo are taken into calculations depending on which is active.
- Ctrl_Byte1: Output byte of reconstructed bits.
- Ctrl_Byte2: Output byte for the shoot signal.



```
if(a!=0){
xx=1;
if(a==1)
x=1;
else
x=0;}
else{
x=0;
xx=0;}

if(s!=0){
yy=1;
if(s==1)
y=1;
else
y=0;}
else{
y=0;
yy=0;}
```

Illustration 30: Data manipulation block diagram.

This shows the formula node that takes the values of the view switch, if the value is “0” then the value of xxo or yyo is “0”. If the value is either “-1” or “1” then the value of xo or xy is “0” and “1” respectively while xxo and yyo are “1”.

It also shows the shoot signal if statement, the “false” case sends a byte of zeros, in the case where it is “true” (Not Shown). A byte of “85” decimal or “01010101” binary is sent.

The block diagram also shows the “bits_byte_string” module which receives different parameters as bits & then transforms them into a byte.

A boolean false signal “0” is sent to the bit_byte_string module to be used for the highest data bit, so the byte value won't exceed “200” as we will discuss the serial module.

▪ Bits_byte_string module:

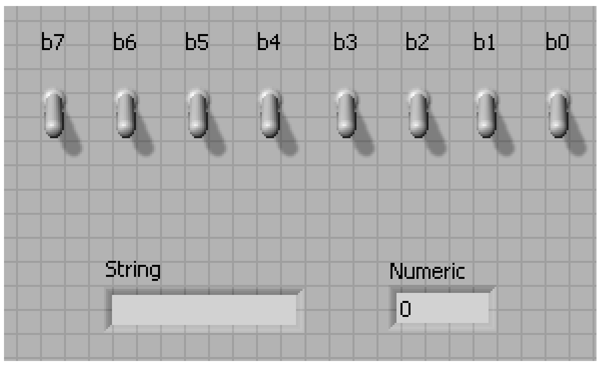


Illustration 31: Bits_byte_string front panel.

The bits_byte_string module is a simple module to construct a byte out of the bits.

Since LabVIEW doesn't have the capability of constructing a byte of of bits for transmission, thins module has been created to solve the issue.

- b0~b7: Represents input bits.
- String: Represents the output ASCII character if available. (String is sent to serial port module)
- Numeric: Represents the output decimal value of the input bits.

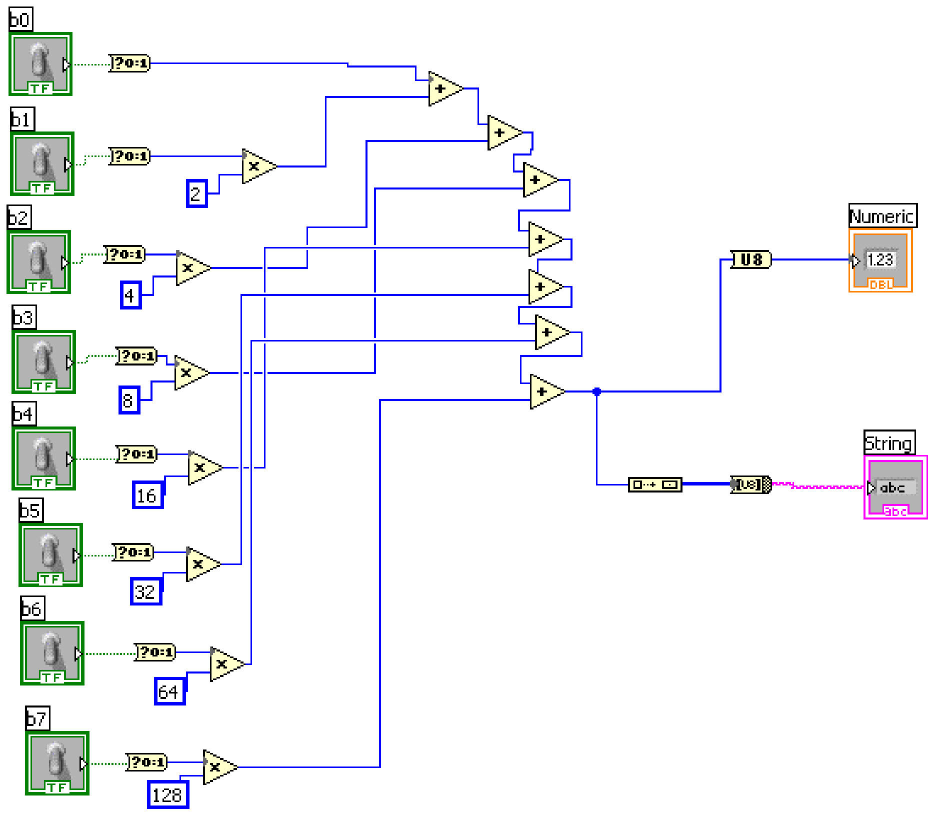


Illustration 32: Bits_byte_string block diagram showing the construction of the ctrl byte 1 of bits.

Illustration 32 represents the construction of a byte from bits. It is done through multiplying bits by the representing value of their positions in the bit.

For example, the first byte is sent as is, the second byte is multiplied by “2” and summed to the previous value, the third byte is multiplied by “8” and summed to the previous value and so on.

It is then represented as a decimal number & string that is sent to the serial module later on.

- **Serial Module:**

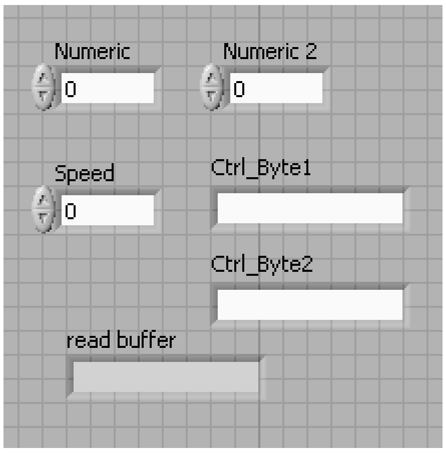


Illustration 33: Serial module front panel.

The serial module initiates the serial port (hardware) of the computer, sets the communication parameters and protocols. It sends a start receiving signal & then it receives the the data bytes being sent from the whole system and sends them through the serial port to the waiting hardware, which is the first micro-controller in this case.

- Numeric: Represents an input that represents the speed of the right wheel servos.
- Numeric 2: Represents an input that represents the speed of the left wheel servos.
- Speed: Represents an input that represents the speed of the servos holding the camera or gun.
- Ctrl_Byte 1: Represents an input that represents the first control byte that contains the data indicating the view switch direction, camera/gun selection, laser status and reset switch status.
- Ctrl_Byte2: Represents an input that represents the shoot byte control.
- Read buffer: Represents the received data from the serial port. (Currently not used, possible uses for future upgrades and two way communications).

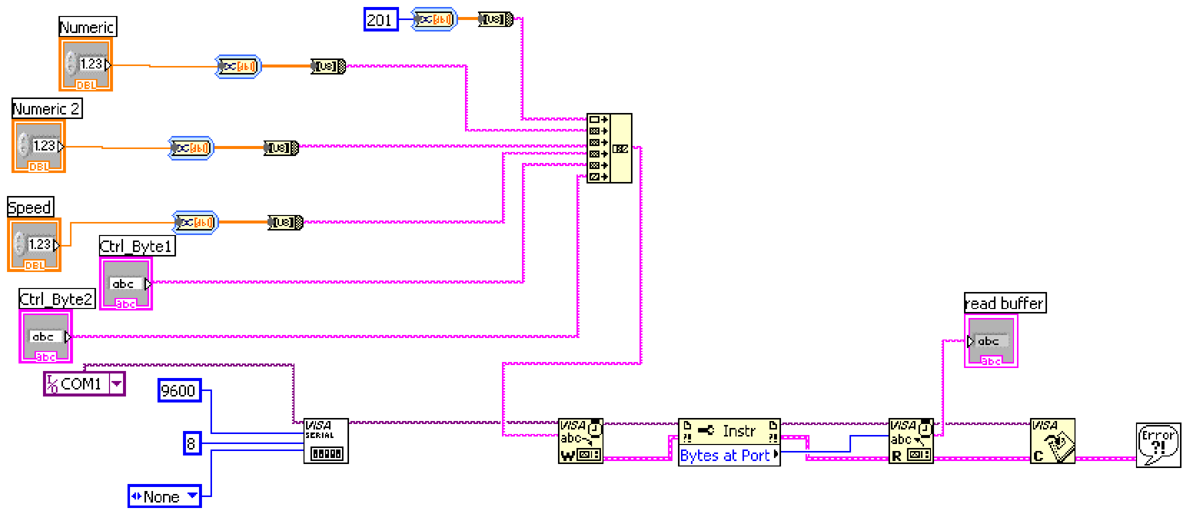


Illustration 34: Serial module block diagram.

The serial port is initiated at COM1, the baud rate is selected to be “9600” bits/s, the data length of “8” bits and no parity.

Other receive from serial port blocks will not be discussed because the function is currently not used.

Since out speed data is limited to 200 (0~200), the start byte should be out of this range, after all we do not want a data to trigger the start receive signal.

In this case the decimal value of “201” as start receiving, remember that in the data manipulation module, a boolean false “0” has been sent to bit_byte_string module to be used as the highest bit, so the byte value can never exceed the “200” value.

Since the VISA module (used for serial communication) in LabVIEW can only use strings for communications, any non-string element should be converted into a string before being sent.

The first byte represents the start receive signal, which is “201” decimal that is converted into a string before being sent.

The second byte is Numeric which represents the speed of the right wheel servos, it is converted from numerical value to string value before transmission.

The third byte is Numeric 2 which represents the speed of the left wheel servos, it is converted from numerical value to string value before transmission.

The fourth byte is Speed which represents the speed of the servos moving the camera or gun, it is converted from numerical value to string value before transmission.

The fifth byte is Ctrl_Byte 1 which has already been described above. It is ready for transmission as it is a string.

The sixth and final byte is the Ctrl_Byte2 which indicates the shoot signal. It is also ready for transmission as it is a string.

Appendix H:

- ### *Micro-controllers Programing:*

  - #### Micro-controller #1:

The first micro-controller will receive the serially sent data from the computer & forwards it to the robot using an RF unit.

Code:

```
'Serial RF TX
' {$STAMP BS2}
' {$PBASIC 2.5}

r VAR Byte
l VAR Byte
speed VAR Byte
dat VAR Byte
shoot VAR Byte

DO
SERIN 1, 16468, [WAIT (201),r,l,speed,dat,shoot]

PULSOUT 0, 1200
SEROUT 0, 16468, [201, r,l,speed,dat,shoot]

LOOP
```

Description:

First few lines are for the design name, micro-controller type and version.

Few variables are being initialized, the ones that have been mentioned before. r for the speed of the wheels of the right servos, l for the speed of the wheels of the left servos, speed for the speed of the servos holding camera and gun, dat for the data byte 1 & shoot for the shoot signal.

The DO LOOP will run infinitely and will keep executing the code inside.

SERIN 1, 16468, [WAIT (201),r,l,speed,dat,shoot]

This initiates port 1 for serial communication at baud rate of “9600” bit/s.

It will wait the start receive signal from the computer which is decimal “201” then it will start receiving the data & storing them into the specified variables.

```
PULSOUT 0, 1200
```

This will sent a pulse with the width of 2µs*1200 to the port 0 where the RF unit is connected, this will enable the receiving RF unit to synchronize it's clock with the RF's transmitter clock in preparation for data reception.

```
SEROUT 0, 16468, [201, r,l,speed,dat,shoot]
```

Will send the data serially to the port 0 (RF transmitter unit) with the same baud rate and the same start receive signal.

- **Micro-controller #2 (Robot):**

The second micro-controller is on board the robot, which receives & executes all the commands.

Code:

```
'Serial RF TX
' {$STAMP BS2}
' {$PBASIC 2.5}

r VAR Byte
l VAR Byte
rr VAR Word
ll VAR Word
speed VAR Byte
dat VAR Byte
shoot VAR Byte

value0 VAR Bit
value1 VAR Bit
value2 VAR Bit
value3 VAR Bit
value4 VAR Bit
value5 VAR Bit
value6 VAR Bit
'value7 VAR Bit

servo VAR Word
servo1 VAR Word
servo2 VAR Word
servo3 VAR Word
servo4 VAR Word
servo5 VAR Word
counter VAR Byte
```

```
'----------------------START------------------------
'Initialize Servos
GOSUB Initialize_Servos:

DO

SERIN 0, 16468, [WAIT(201), r,l,speed,dat,shoot]

rr = 1500 - 650 - r
ll = 650 + l

PULSOUT 12, rr
PULSOUT 13, ll
PULSOUT 14, rr
PULSOUT 15, ll

GOSUB Byte_Disassembly:

'hat x gun
IF((value5=1)AND(value2=1)) THEN
IF((value6=0)AND(servo1<=1050)) THEN
servo1 = servo1 + speed
ELSEIF((value6=1)AND(servo1>=250)) THEN
servo1 = servo1 - speed
ENDIF
ENDIF
'hat y gun
IF((value3=1)AND(value2=1)) THEN
IF((value4=0)AND(servo2<=1050)) THEN
servo2 = servo2 + speed
ELSEIF((value4=1)AND(servo2>=250)) THEN
servo2 = servo2 - speed
ENDIF
ENDIF
```

```
'hat x cam
IF((value5=1)AND(value2=0)) THEN
IF((value6=0)AND(servo4<=1050)) THEN
servo4 = servo4 + speed
ELSEIF((value6=1)AND(servo4>=250)) THEN
servo4 = servo4 - speed
ENDIF
ENDIF
'hat y cam
IF((value3=1)AND(value2=0)) THEN
IF((value4=1)AND(servo5<=1050)) THEN
servo5 = servo5 + speed
ELSEIF((value4=0)AND(servo5>=250)) THEN
servo5 = servo5 - speed
ENDIF
ENDIF

'Laser
IF(value1=1) THEN
HIGH 6
ELSE
LOW 6
ENDIF

'Shoot Condition + Safety

IF(shoot=85)THEN
FOR counter=0 TO 20
PULSOUT 3,290
'-----HERE---
PULSOUT 1,servo1
PULSOUT 2,servo2
PULSOUT 4,servo4
PULSOUT 5,servo5
```

```
PAUSE 10
NEXT
FOR counter=0 TO 10
PULSOUT 3,600
PAUSE 20
NEXT
ENDIF

IF(value0=1) THEN
GOSUB Initialize_Servos:
ENDIF

'Give Servo Signal
PULSOUT 1,servo1
PULSOUT 2,servo2
PULSOUT 4,servo4
PULSOUT 5,servo5


LOOP

'-----------------Subroutins-------------
Initialize_Servos:
servo=650
servo1=650
servo2=650
servo3=650
servo4=650
servo5=650
FOR counter=0 TO 50
PULSOUT 1,650
PULSOUT 2,650
PULSOUT 3,650
PULSOUT 4,650
PULSOUT 5,650
```

```
PAUSE 20
NEXT


RETURN

Byte_Disassembly:
temp VAR Byte
'value7=dat/128

'IF(dat>=128) THEN
'temp=dat-128
'value6=temp/64
'ELSE
value6=dat/64
'ENDIF

IF(dat>=64) THEN
temp=dat-64
value5=temp/32
ELSE
value5=dat/32
ENDIF

IF(dat>=32) THEN
temp=dat-32
value4=temp/16
ELSE
value4=dat/16
ENDIF

IF(dat>=16) THEN
temp=dat-16
value3=temp/8
ELSE
```

```
value3=dat/8
ENDIF

IF(dat>=8) THEN
temp=dat-8
value2=temp/4
ELSE
value2=dat/4
ENDIF

IF(dat>=4) THEN
temp=dat-4
value1=temp/2
ELSE
value1=dat/2
ENDIF

IF(dat>=2) THEN
value0=dat-2
ELSE
value0=dat
ENDIF

RETURN

END
```