

Robot Wars

Project Report

By
Daniel Collins,
2007624

BSc (Hons) in the Internet of Things,
2nd Year Project Report, 2018,
Department of Science and Computing,
Waterford Institute of Technology

Title: Robot Wars Project Report

Author: Daniel Collins, 20076240

Supervisor(s):

Jason Berry

Abstract

This report will describe an embedded system attached to a “Robot”. It will go through the procedure involved in getting the PCB’s to interact and work from/with each other using a CAN network, the installation of the boards to the “Robot”, the challenges involved with the project etc.

This report will also include content from Semester 1 of Second Year, where a PCB Board was designed from scratch, documenting its creation, population of the board with components, and verification of the boards operation through various tests.

The Final outcome of the Project was a functioning Robot, controlled using three separate self-designed PCB Boards connected by a CAN Network and controlled by the most part by an application called “RoboRealm”.

Table of Contents

Abstract.....	3
Introduction	6
Hardware Description	7
Power Section	7
PIC Section	8
Clock Section	9
Buttons Section	9
RS232 Section.....	10
Indicator LED Section	10
LCD Section	11
Digital I/O	11
Analog I/O	11
CANBUS Section	12
PCB Design & Manufacture.....	13
Schematics	13
Netlist & PCB Layout	13
Routing.....	14
Printing.....	15
Applications of a PCB	15
Medical Devices	15
Consumer Electronics	15
Industrial Applications	16
Project Module Application	16

Overview	16
Hardware Description	17
The Bot	17
Joystick	18
Sensors	19
CAN Cable.....	20
Software Overview.....	21
PIC Downloader	21
Hyperterminal	22
RoboRealm	23
Conclusion.....	30
References	31

Introduction

In this report on the Project Module of the BSc (Hons) in Internet of Things course, I will provide an in-depth review of the PCB Board, from its initial design on the Eagle Application to the final, fully working PCB. I will then proceed to explain the implementation of these PCB Boards onto a “Robot” and the applications achieved through the “Robot”

I shall explain what each sections role in the use of the board is, how the board works and what the board could potentially be used for.

Firstly, a Printed Circuit Board (PCB) is a board that holds physical components by “Soldering” them onto the board itself. It allows the transfer of signals and power between these physical devices. The connections are made by small “tracks” of copper wire printed onto the board. As the solder holding the components onto the board is metal, it acts as a conductor and links the tracks with the components.

Hardware Description

On my own PCB, there are nine major sections, using a wide variety of physical hardware components. Below, I will discuss each section and its functionality.

Power Section

This section provides the power to the entire board. Any components that require power are linked to tracks that are all connected to the Power section.

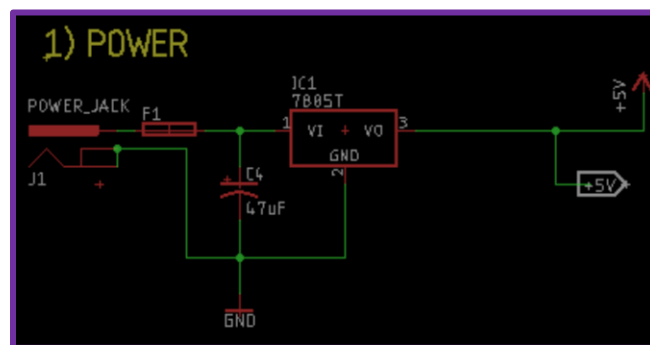


Figure 1.0

It contains a DC Power Jack, attached to a fuse, for safety, and running through a Voltage Regulator. The Regulator is there as mains voltage in Ireland is about 12V, but that would be too much for the board to handle. There is also a capacitor connected to the Voltage Regulator.

PIC Section

A PIC (Figure 2.0) is a type of microcontroller, which essentially means it is a microcomputer responsible for the operation of the board. It governs what signals are sent where and at what time.

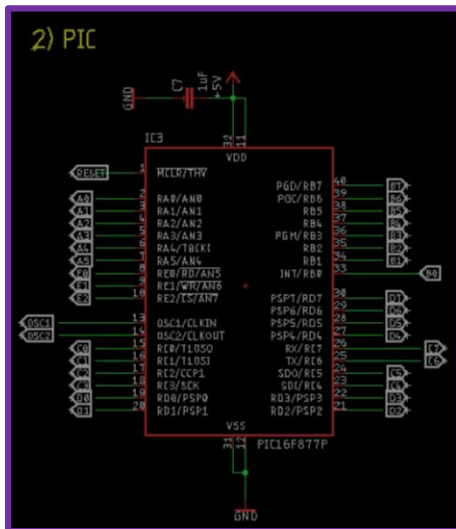


Figure 2.0

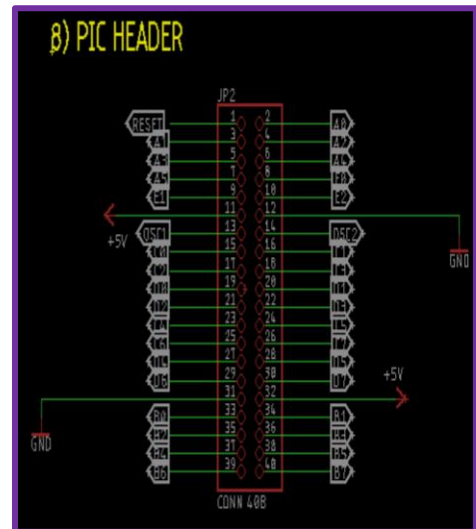


Figure 2.1

Clock Section

The speed of a CPU is determined by the Clock-Cycle, or the amount of time between two pulses of an oscillator [1]. Generally, the higher the number of pulses a second, the faster the CPU can operate.

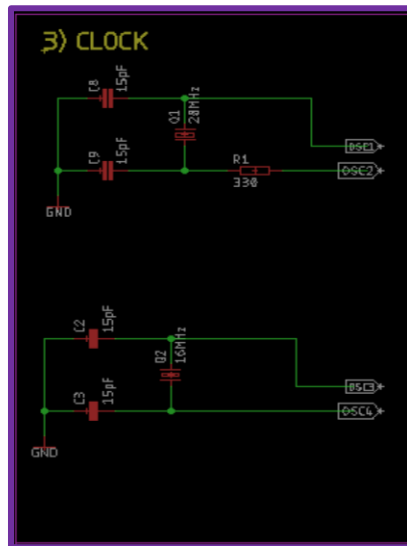


Figure 3.0

The clock circuit consists of a 16MHz and a 20MHz clock.

Buttons Section

This section refers to the implication of the Push-Button Switches installed on my PCB.

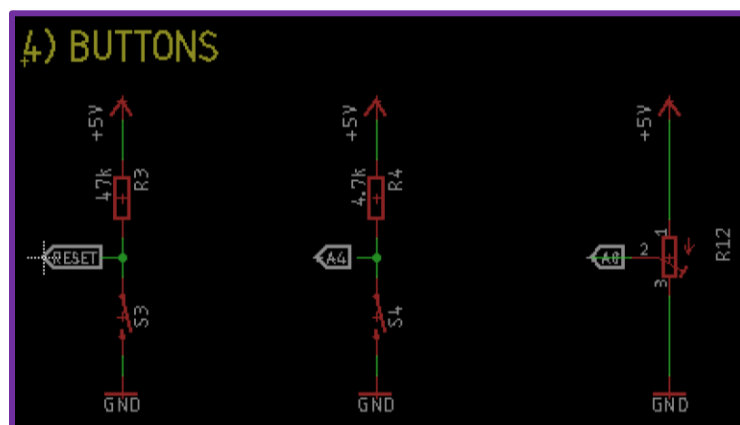
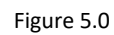
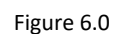


Figure 4.0

This section handles all types of Serial Communication. RS232 stands for Recommended Standard-232. [2]



There are 6 LED's incorporated in my PCB, all of which are indicators something has happened. i.e. if the Power LED is illuminated, it is a signal that there is power reaching the board.



LCD Section

For my PBC, I have installed a backlit LCD display unit. This section holds all the necessary connections needed to link the display with my board.

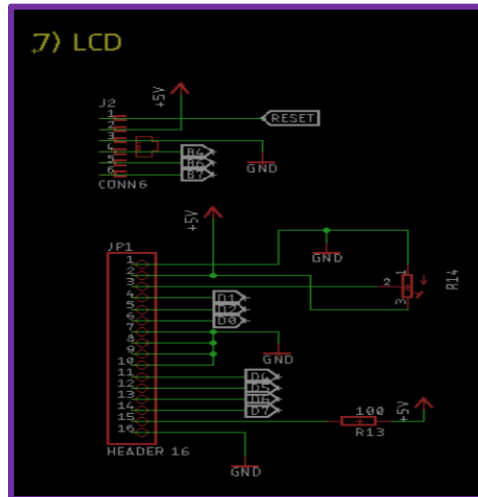


Figure 7.0

Digital I/O

Digital I/O stands for Digital Input and Output. Digital Inputs allow a microcontroller to detect logic states, and Digital Outputs allow a microcontroller to output logic states. Each Digital I/O can be in one of three states; Input, Output-High or Output-Low.

Analog I/O

Analog signals are variable, which means they have multiple states. Analog input signals can represent items such as speed or distance. Analog output signals are also variable and can be used for such things as moving an actuator to a desired position.

CANBUS Section

The CANBUS Section (Controller Area Network BUS) is used to allow microcontrollers and devices to communicate with each other without the need of a host Computer.

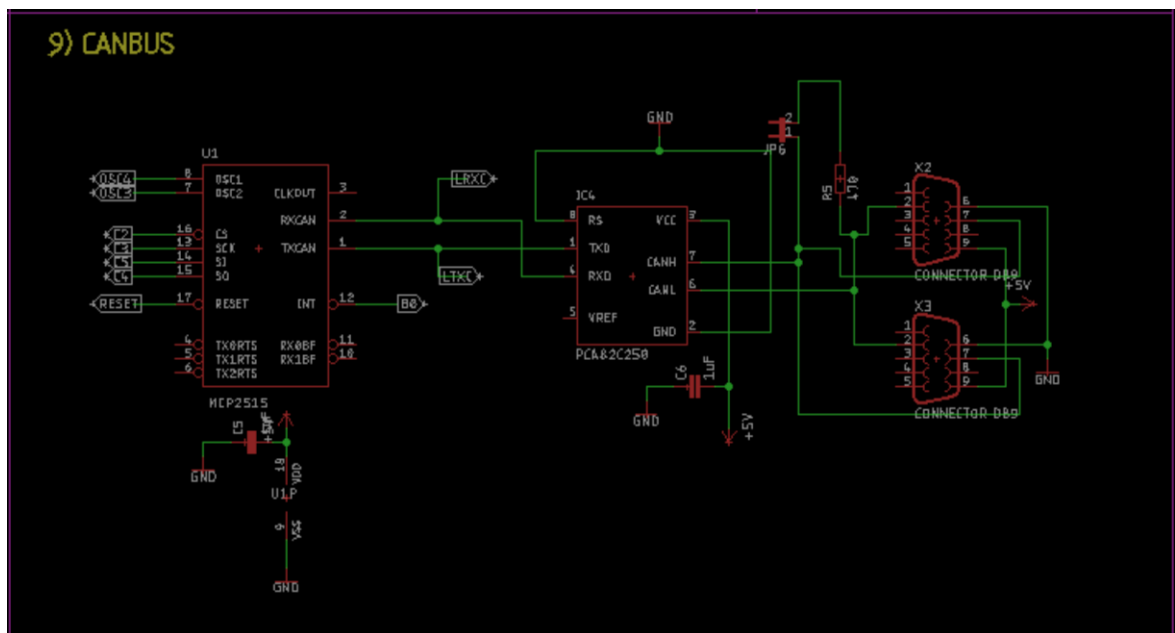


Figure 8.0

CAN is a message-based protocol. It allows multiple devices to communicate with each other without the need for complex wiring.

PCB Design & Manufacture

When designing the PCB, we used an application called Eagle. We were designing our PCB's alongside Engineering Students, however none of us had used Eagle before so it was a very good experience.

Schematics

We were presented with Schematics which were basically circuit layouts for our PCB from our Lecturer. These Schematics outlined how all the components were to be wired together. Producing the matching circuits on Eagle was tricky at first, due to our lack of knowledge in this area, however, we were quickly able to get to grips with it. There were still a few issues that were not spotted at first, such as some of the students having incorrect footprints for the components, but these issues were promptly caught and corrected.

Netlist & PCB Layout

A Netlist is a description of all the connections in a circuit. It consists of all the electrical components and their respective connections. Once we had produced the Schematics, we then generated a Netlist for the PCB. From the Netlist we were able to check all our connections and ensure there were no broken connections. With this complete, we moved on to assembling a digital PCB layout (Figure 9.0), specifying where on the board each component would be placed. This took quite some time as it was a tight squeeze to fit all the components on the board. There were also some guidelines that needed following, such as having all components from one circuit section as close as possible to each other.

Printing

Once all the above was complete, it was time to send off the Eagle-generated files to our designated PCB Manufacturer, BETA, who then printed our boards and sent them back to us.

Applications of a PCB

At the heart of electronics live PCB's, whether we are talking about smartphones, kitchen appliances, or large Industry machines. PCB's are used in almost every industry and are firms are continuously finding new ways to incorporate them.

Some of the applications of PCB's are listed below;

Medical Devices

Nowadays, a large majority of medical equipment is electronic, such as Heart Monitors, Blood Pressure Monitors, Internal Devices (Pacemakers) and Scanning Equipment. Because of the responsibility of the PCB's in such devices, medical PCB's are held to a much higher standard than other PCB's [3].

Consumer Electronics

Consumer Electronics refer to the devices we use every day, from phones to washing machines to computers. Due to the amount of consumer electronics being produced, and equal amount of PCB's must be produced. Due to this, there is a challenge associated with keeping all PCB's produced at the same standard, and as a result, PCB manufacturers have strict quality control measures put in place to ensure all electronics work as they should.

Industrial Applications

PCB's are often used in high-powered industrial equipment. These PCB's must be able to handle the harsh conditions often seen by Industrial Equipment. This could be anything from chemical exposure to extreme vibrations to rough handling. Industrial Applications of PCB's can be anything from a simple Hand-Drill, to solar power cogeneration equipment to equipment used to measure pressure and other variables.

Project Module Application

Overview

For this Module, we were required to connect multiple of our PCB boards to a deconstructed kids version of a Segway. We were required to gain control of the existing motor controller, install multiple sensors on the bot, implement a vision system using "RoboRealm" and then sonify the entire bot. The Bot's build and installation of the sensors was delegated to Darren and I, Loti was in charge of the Vision System and Seamus was in control of the Sonification. We used three boards for this build.

Hardware Description

There were few extra Hardware components used once the Bot was acquired, however I will explain the main extra components.

The Bot

The Boards were installed on a deconstructed Feber Dareway, with the handles removed.



This machine came with a built-in motor controller for controlling the two wheels, and this was easily hacked to provide us with full access and control over the motors.

Joystick

The Joystick was used for the manual driving. We connected the Joystick to the laptop and mapped certain variables to its movement.



Sensors

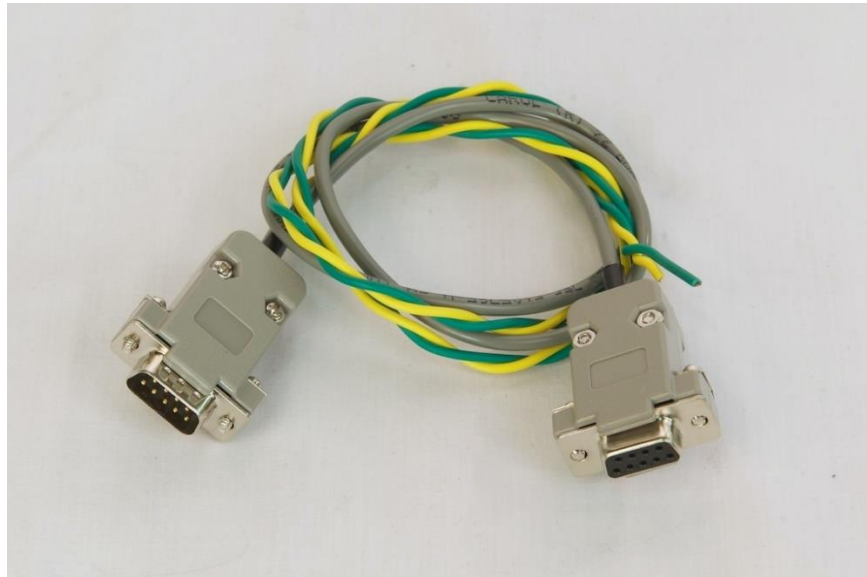
We used four distance sensors on our build. Darren installed two sensors on the front corners of the bot and two sensors on the back corners of the bot.



The function of these sensors was to be able to anticipate any objects coming towards the bot. These four sensors were connected back to the Inputs Board.

CAN Cable

A CAN Cable is used to link different components on the same CAN Network together, so they can communicate with each other.

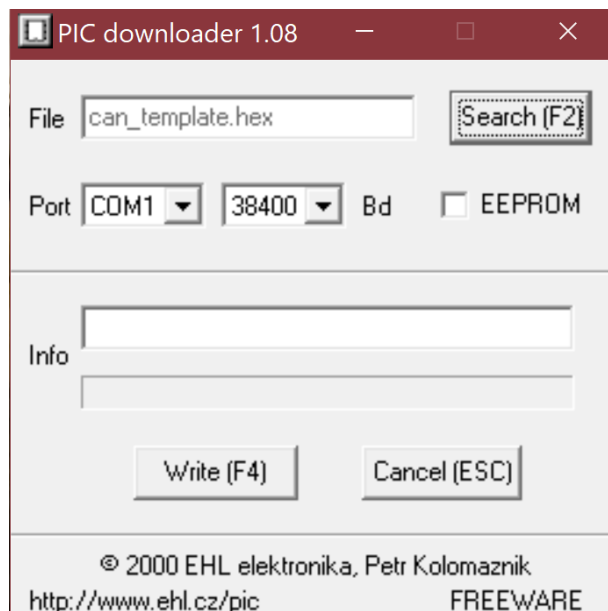


Software Overview

Once the Boards had been implemented on the Bot, they were programmed with some pre-produced code. This section will describe how this was achieved, using various applications. It will also look at “RoboRealm”, the application used for most of the functionality of the Bot, and “Hyperterminal”, a Terminal Emulator used mostly for debugging purposes.

PIC Downloader

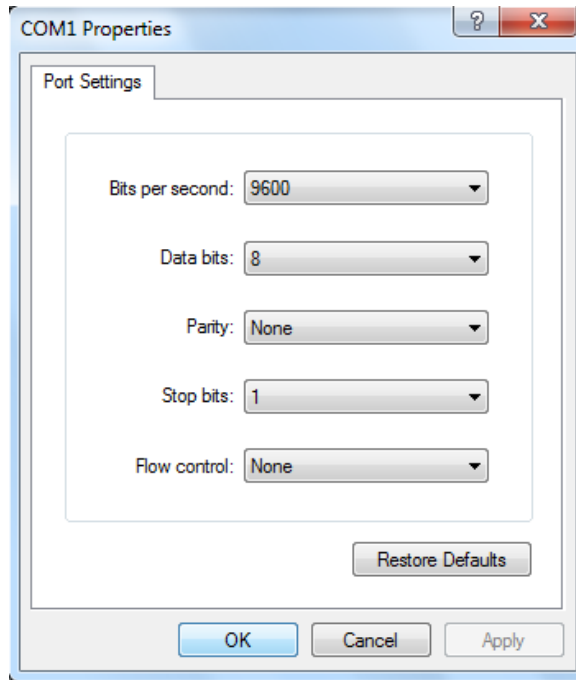
After the pre-produced code was compiled in the C programme, it was transformed to a .hex file and uploaded to the PIC of one of the boards. This was done using PIC Downloader, a software application as seen below.



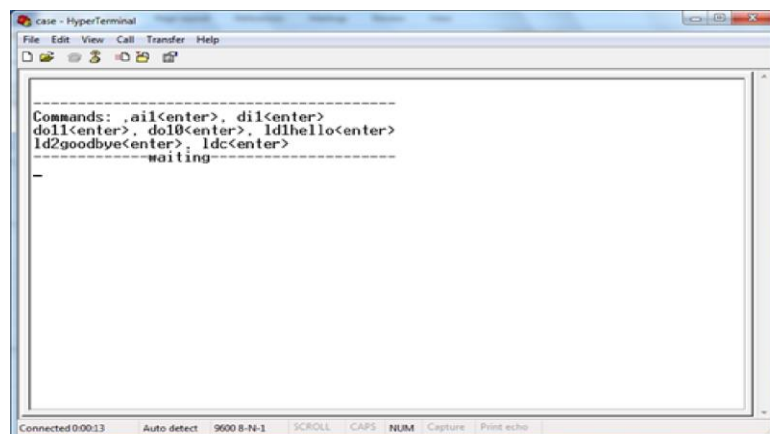
For the board to be able to receive the file, it must be Reset which gives the board time to load up the file.

Hyperterminal

Hyperterminal is a piece of software which is used to read and write data to/from the PIC. Below are the multiple Port Connection Settings available. We require the “Bits Per Second” tab to be 9600, the “Data Bits”, “Parity”, and “Stop Bits” remain at their default values, and the “Flow Control” is required to be “None”.

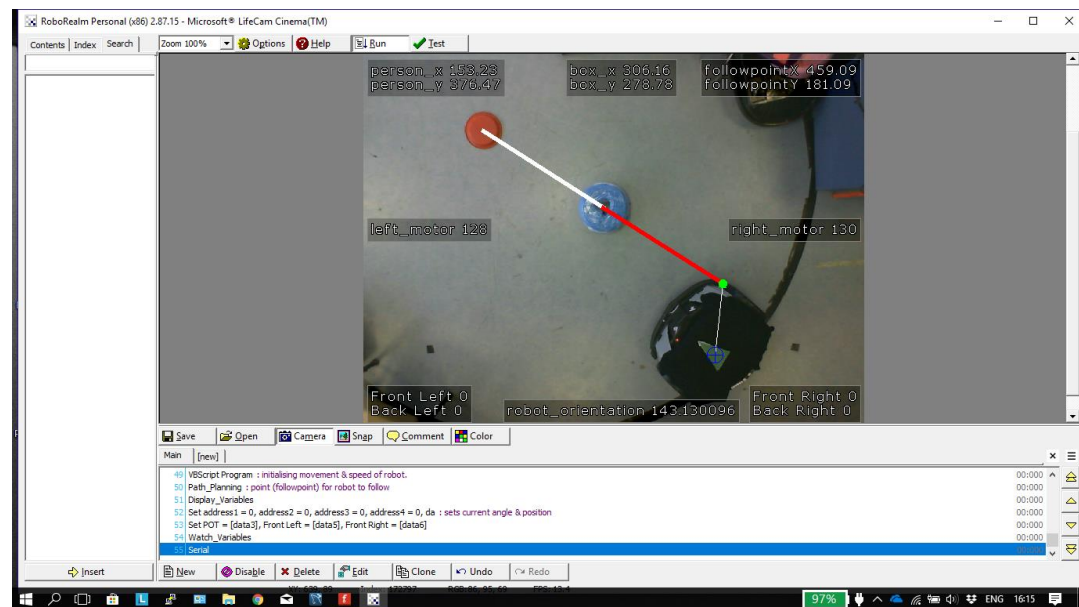


Once a connection to the Port is established, we must put the Board into RX:2 mode, which allows the board to send out messages and data.



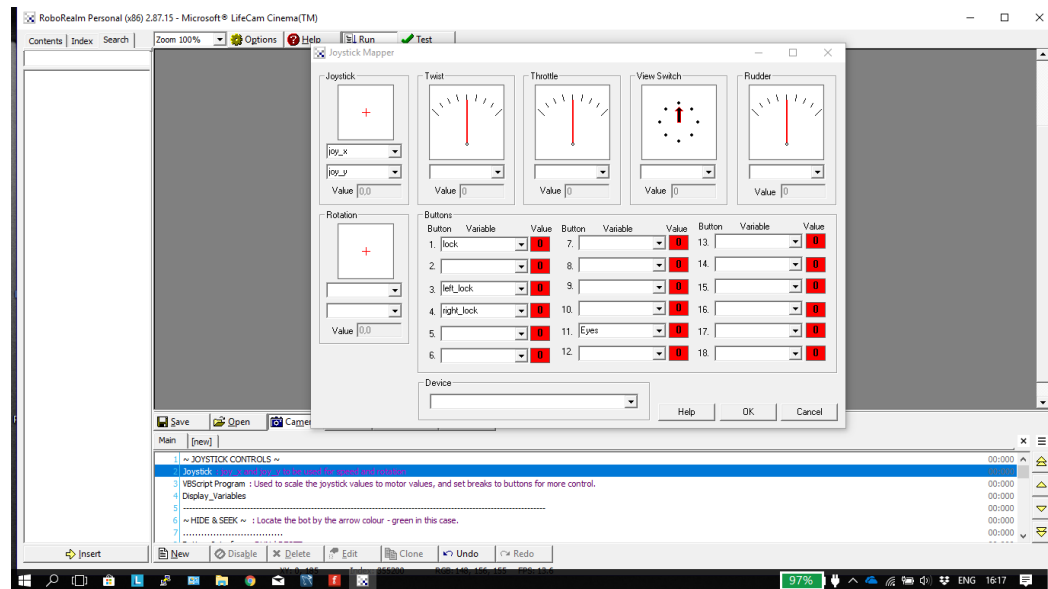
RoboRealm

RoboRealm is the main application used for implementing any kind of functionality with the bot or “Getting the bot to do something”. It has the option of connecting a camera, which was ideal for the vision system. It also has the option of connecting a Joystick and mapping certain functions to each button or direction of the Joystick which provided a platform for the manual driving of the Bot. Below is a general idea of what RoboRealm looks like.

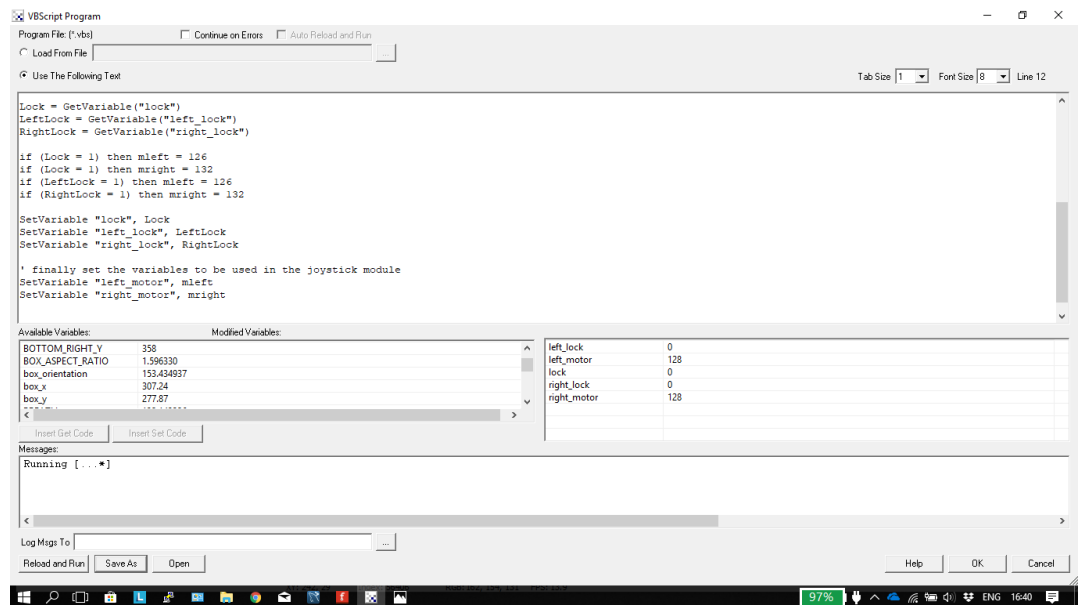
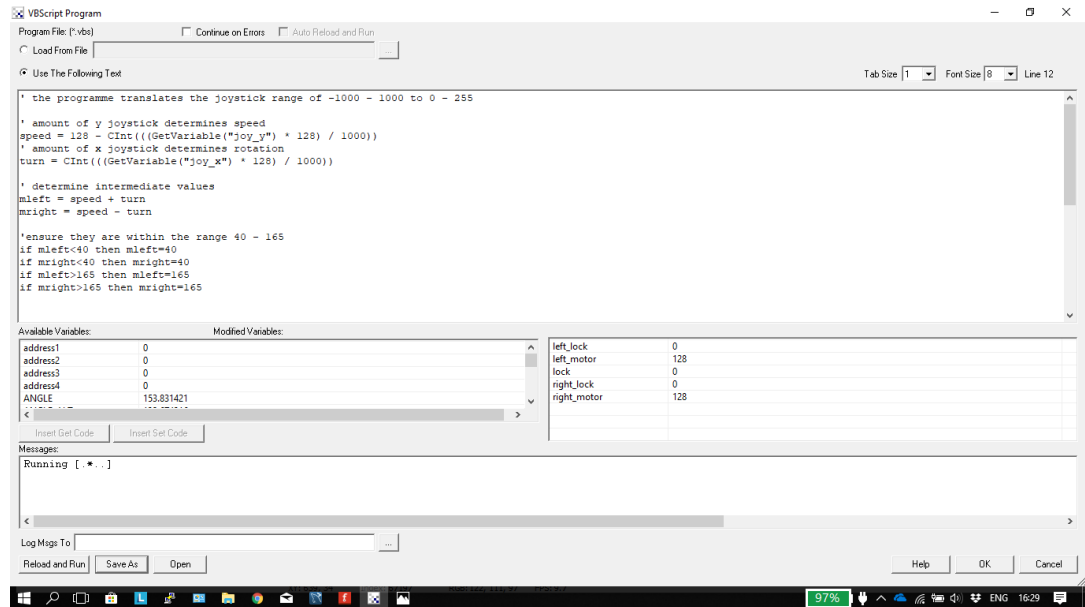


RoboRealm was used mostly for the vision system that was implemented on the Bot. The Vision System involved linking the RoboRealm application to a camera, which used the camera feed to do certain methods.

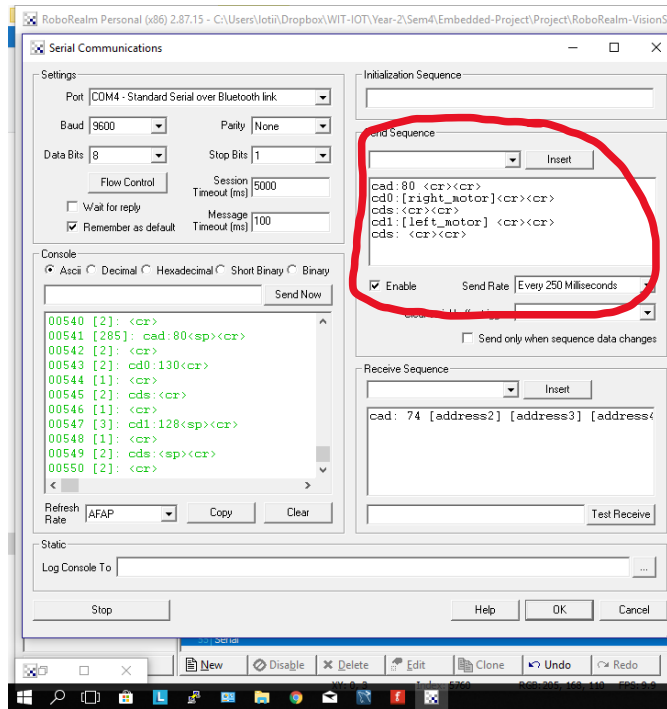
Before the vision system could be installed, however, we needed to get the manual drive working on the Bot. This involved connecting a joystick to the Laptop and mapping the motor controller values to the different actions performer with the Joystick.



This is the Joystick Module from RoboRealm. It gives us options to assign variables and corresponding values to all the buttons on the joystick, as well as giving us the X and Y position of the joystick. These X and Y values were manipulated to send corresponding values to the motor controller, thus driving the motors. Loti also implemented “Left Lock”, “Right Lock”, and “Full Lock” buttons, which, when pressed, completely stopped either the Left Wheel, Right Wheel, or Both Wheels. While the values received from the X and Y positions of the joystick were extremely valuable and enabled us to do a lot with the Bot, the values were not within the range that we needed them to be. Therefore, there were some calculations that needed to be performed on these raw values to convert them to usable ones. Below are screenshots of these calculations.

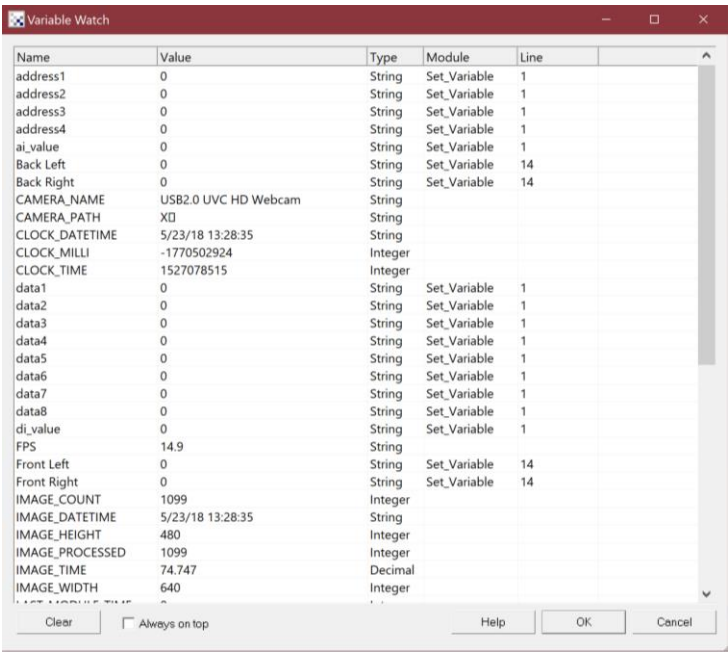


The Serial Module is one of the most important modules in RoboRealm. It allows RoboRealm to send and receive data to and from the Bot. For example, when the Joystick is moved, changing the values for the motor controller would be of no use unless we are able to send those values out to the Bot. The “Send Sequence” in the Serial module allows us to do just that.



This bot of code essentially gets the values stored in the variables “right_motor” and “left_motor” and sends the to the Left motor and Right motor on the Bot.

The Receive sequence also proved to be of some importance with the Bot. Here, the Serial Module receives all the messages that the Bot is sending out. What we needed to do was pull out the relevant values we were looking for, namely the four sensor values. However, the Bot does not know what values/variables we are looking for, so it sends out the full message. To overcome this and be able to select what data we want to use, we basically used buffer variables. These are new variables we created and populated with the corresponding data received from the Bot’s messages. From these new variables, we are then able to select the correct variables with the data we are looking for.

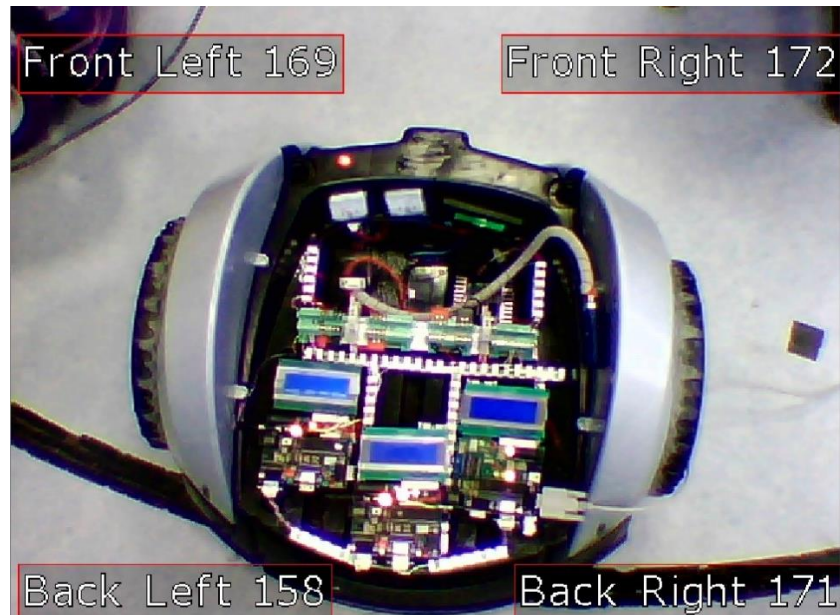


The screenshot shows a 'Variable Watch' window with a table of variables. The table has columns for Name, Value, Type, Module, and Line. The variables listed include address1 through address4, ai_value, Back Left, Back Right, CAMERA_NAME, CAMERA_PATH, CLOCK_DATETIME, CLOCK_MILLI, CLOCK_TIME, data1 through data8, di_value, FPS, Front Left, Front Right, IMAGE_COUNT, IMAGE_DATETIME, IMAGE_HEIGHT, IMAGE_PROCESSED, IMAGE_TIME, and IMAGE_WIDTH. The values for these variables are shown in the 'Value' column. The 'Module' column shows 'Set_Variable' for most variables, and the 'Line' column shows the line number where the variable was defined.

Name	Value	Type	Module	Line
address1	0	String	Set_Variable	1
address2	0	String	Set_Variable	1
address3	0	String	Set_Variable	1
address4	0	String	Set_Variable	1
ai_value	0	String	Set_Variable	1
Back Left	0	String	Set_Variable	14
Back Right	0	String	Set_Variable	14
CAMERA_NAME	USB2.0 UVC HD Webcam	String		
CAMERA_PATH	X:\	String		
CLOCK_DATETIME	5/23/18 13:28:35	String		
CLOCK_MILLI	-1770502924	Integer		
CLOCK_TIME	1527078515	Integer		
data1	0	String	Set_Variable	1
data2	0	String	Set_Variable	1
data3	0	String	Set_Variable	1
data4	0	String	Set_Variable	1
data5	0	String	Set_Variable	1
data6	0	String	Set_Variable	1
data7	0	String	Set_Variable	1
data8	0	String	Set_Variable	1
di_value	0	String	Set_Variable	1
FPS	14.9	String		
Front Left	0	String	Set_Variable	14
Front Right	0	String	Set_Variable	14
IMAGE_COUNT	1099	Integer		
IMAGE_DATETIME	5/23/18 13:28:35	String		
IMAGE_HEIGHT	480	Integer		
IMAGE_PROCESSED	1099	Integer		
IMAGE_TIME	74.747	Decimal		
IMAGE_WIDTH	640	Integer		

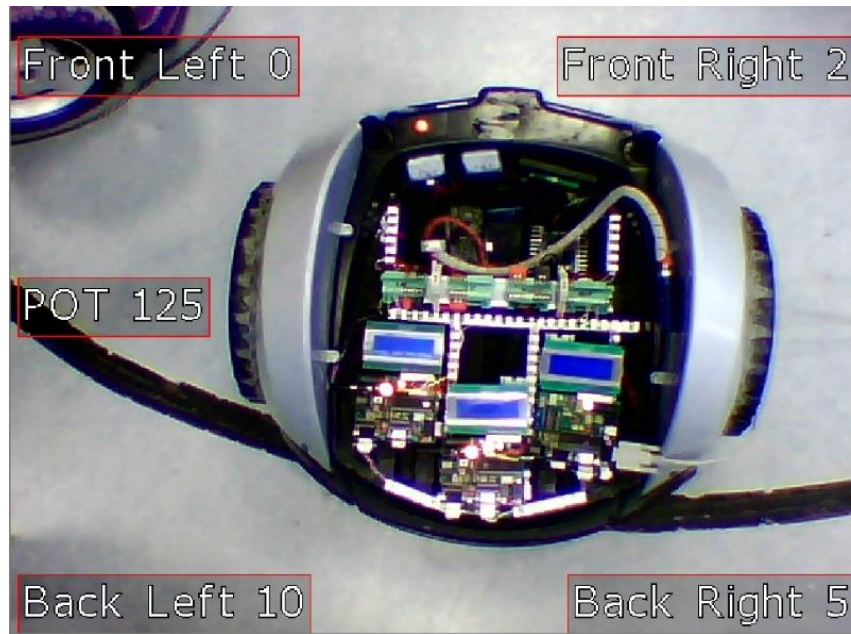
The “data1”, “data2”, “data3” etc. values are all the buffer variables that would be populated with the variables from the Bot’s message. If we were looking for the value of the front left sensor, and that was the fourth data bit of the Bot’s message, we could just pull the value from the newly populated “data4” variable and this would be the corresponding value.

The values coming in from the sensors however were effectively juts numbers and did not mean anything to the average person.



Raw Values from Sensors

Darren performed a calculation of these values, getting the maximum value and the minimum value, while also noticing that once the minimum value is reached there seems to be a “Dead Spot” where the values start to go up again. Once Darren had found the minimum and maximum values, he mapped these onto cm values, so we had an idea of how far away things were from the Bot. This involved using some complex math equations, and once the conversion was performed, the values displayed made more sense.



Converted Values from Sensors

This was essentially a breakthrough with RoboRealm, as from my understanding, this was the first time we had been able to receive and use data from the Bot. However, this breakthrough came near the end of the project and as a result we were not able fully show the potential applications this provided.

Conclusion

The aim of this project was to create a working “Robot” essentially from scratch, using our own PCB boards. “A robot is a machine—especially one programmable by a computer— capable of carrying out a complex series of actions automatically”. I feel like our attempt at creating a Robot meets this description so therefore we have been successful. The Bot created was able to be manually driven, autonomously driven, and was able to perform different actions based on the environment it encountered (Vision System). We were asked to create a Bot that would be both challenging and intriguing to the younger generation and I feel like we achieved this. We delivered the Bot to the specifications set out to us, although sometimes we fell just short (Sonification) and sometimes we pushed past what was expected of us (Vision/Receiving from Bot).

Overall, I feel as though there was not any major flaws within our team. I thought having two second year I.O.T students mixed with two third year students would cause some challenges but instead of clashing with each other, we complimented each other well. The communication between all the team members was a quintessential part of our success in this module.

I would have liked to have had more time with the Bot and with my team to see how far we could have brought the Bot, as I feel as though we have achieved quite a lot in a short space of time from the limited knowledge we began with.

References

- [1] <https://www.quora.com/What-is-clock-cycle-machine-cycle-and-instruction-cycle-in-a-microprocessor>
- [2] <https://en.wikipedia.org/wiki/RS-232>
- [3] <https://www.pcbcart.com/article/content/PCB-applications.html>
- [4] <https://en.wikipedia.org/wiki/Robot>