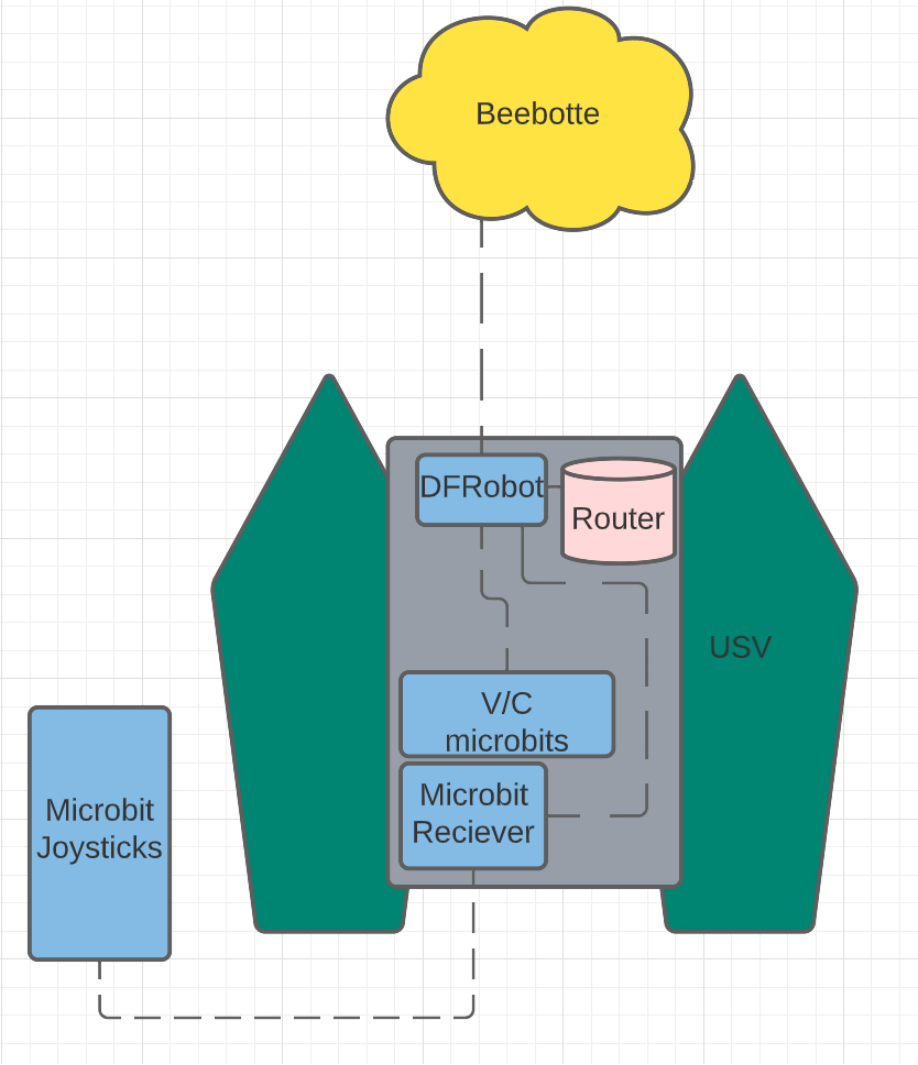
Project 4

Adam O'Brien
20093460

Applied Computing (IOT)

Waterford Institute *of* Technology

TABLE OF CONTENTS



INTRODUCTION

Project 4

Project 4 is one of my Modules that is Part of my Course Applied Computing (Internet of Things). The aim of the module is to implement the baseboard that was built in Semester 3 and build something with it.

The Microcontroller that was built uses a PIC16F877A-I/P microchip.

The team for this project included second and fourth year IoT students third year electronic engineers and two masters' students.

Everybody was assigned a different job for this project according to their skillset just like how a real workplace would operate,
For example, students who studied IoT were a lot more involved with the coding side of the project whereas the students studying electronic engineering put their focus into Designing, inspecting, testing, and updating electronic systems.

This Semesters theme was to build a USV (Unmanned Surface Vehicle),

The way the USV would work is it would be docked by a river and a drone would survey a body of water, if a person was to enter the water and start struggling the drone would then fly over to the person and call the USV to attempt to save the drowning person.

This Semester the team has successfully build a working USV with IoT functionality.

Project Overview

The project itself consists of many working parts; these include:

- The USV itself as a vehicle
- A drone to survey any body of water
- Cloud connectivity (data sent to and from IoT Platform)
- Radio Network
- Ease of use

USV:

The Unmanned surface Vehicle is a converted fishing pontoon with a JEEP roof rack mounted on top to keep electronics dry and provide a sleek and unique look.

The main electronics are

- Two 12Volt waterproof Batteries
- Four T200 water-cooled Thrusters
- A Multiplexer to switch from different control channels (RC -> Micro bit -> Cloud Drive)
- BBC Microbit

Thrusters

The USV currently has four T200 Thruster BlueROV2.

The operating Voltage of these motors are 7-20 volts, source:
(https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/)
The first river test was done at 12 volts total.
These Thrusters have been pushed to 24Volts

These thrusters were the top choice for the following reasons

- Water Cooled motors
- Compact size
- Easy mounting
- Ease of use (Works with the same PWM signals as normal continuous servos)

Batteries

The batteries used for the USV were fully waterproof, this was essential as if these batteries did get water damaged the battery could die or at worst it could leak or even explode and all the electronics in the USV would be destroyed.

Multiplexer

The multiplexer allows for easy switching from RC control to microbit control. There is a multiplexer on the USV and the switch to change from RC to microbit is located on the RC controller.

BBC Microbit

The BBC Microbit is a small micro-controller mainly used in schools to tech children the basics of coding.

These microcontrollers may be small, but they offer an extremely wide range of features such as

- Radio, Serial, Bluetooth Comms
- Can program the PIC16F877A-I/P when paired with the Kitronik Servo driver board
- Built in variables such as Pitch, Compass, Temperature, light level, acceleration that can be easily implemented in projects.
- It Has its own Math and logic libraries for more complex programming.

The USV is also able to connect to the cloud through a router that is placed inside the roof rack, this then connects to a DFRobot Microbit cloud expansion board which sends all data to BeeBotte.

There is also a vision system that has to be implemented physically but all of the background work has been tested

Drone:

The Drones job is to survey a body of water and to look out for anything out of the ordinary, for example a person drowning, the drone will then fly over to the person in trouble and communicate with the USV to start a rescue mission, The USV will then sail over to the individual in need of assistance and hopefully save them.

There has also been a microbit extension made for the Tello Drone by Victor Alikberov a masters student that is part of the team.
This will allow for children and people with little experience in coding to send the tello drone on missions.

Cloud Connectivity:

Cloud Connectivity was mostly covered by the IoT students on the project this feature was a great addition to the overall project as it included cloud drive commands and the recording of USV data (for example current going to motors)

Radio Network:

The radio network was implemented using BBC micro bits, The bulk work of this radio network was designed by myself. The reason for the micro bits implementation in this project is because of its ease of use, they allow for fast prototyping in a very friendly code-readable environment. The radio networks place in the project was to send data around the microbit network from one microbit to another and then use the received data to push onto the cloud or control part of the USV,
For example, the Joystick microbit sends a tilt value to the receiver microbit and then the receiver microbit turns the USV's motors according to the angle the Joystick is tilted, so the greater the tilt the faster the motors would go. Every microbit radio network code will be listed in the My Role in the Project Section.

Ease of use:

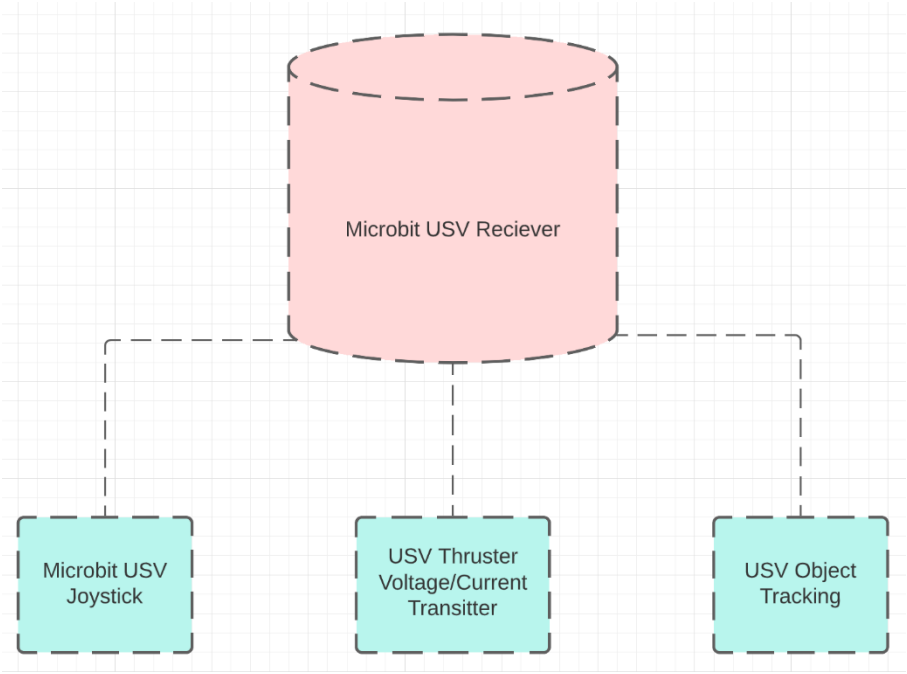
Ease of use was a very important step for this project because this project will continue to be worked on so new team members must be able to understand the inner workings quickly and efficiently,
For this project one of the end goals was to allow children to send the USV out on missions and the micro bit interface is the best option to get kids interested and inspired by this work.

My Role in the Project

My role in the project was to complete the following tasks

- Build a radio network for the USV to allow for joystick movement
- Connect micro bit to MQTT Services
- Use the radio network to achieve autonomous movement
- Testing micro bit interface with the baseboard

In order to achive an reliable connection between the microbit and the USV I used the microbits built in radio makeblocks.



USV reciever – The USV Reciever microbit listens for to all of the other microbits on the radio system.

Microbit Joysticks – The microbit Joystick code is very short. All this does is it takes the built in pitch(roll) of the microbit and maps it to a value that gives the desired pulse width modulation for the motors to move. Also if these microbits loose communicaion with the reciever the reciever will set the motors to the stop position to prevent the USV from crashing

Thruster Voltage and Current – The Thruster Voltage and current microbits take in the voltage and current of the batteries on the USV and do several calculations to give us an accurate reading of the current and voltage. This is then sent to the DFRobot board through radio comms and then pushed to the Beebotte network to be viewed on an online Dashboard.

Radio Network Signal Names

- "left" – moves the left motor whatever value it is paired with (60 – 120)
- "right" – moves the right motor whatever value it is paired with (60 – 120)
- "go" – moves both motors whatever values they are paired with (60 – 120)
- "stop" – stops both motors
- "rtc" – right thruster current
- "ltc" – left thruster current
- "rtv" - right thruster voltage
- "ltv" – left thruster voltage

Radio Network Signal Values

The radio network works be sending values along with a name so the reciever can apply that value to the name it is paired with.
The motors work between a pulse width of 1.8 –> 1.2
1.8 = 120 = full speed forward
1.5 = 90 = stop
1.2 = 60 = full speed backwards

What this means is that if I write some microbit code as simple as this.

Then the reciever will receive that piece of code and make the left and right motor move at full speed “forever”.

This makes it incredibly easy to make the USV move autonomously because you could just add pause makeblocks to make it “go” for 5 seconds then make it turn left by sending “right” and make it turn right by sending “left”.

IoT Cloud –

The IoT cloud was a big part of this project because without it this would just be an RC USV. All of these features listed so far have had the data that is being sent/received uploaded to a cloud service. The reason it has been sent to these cloud services is so dashboards can be made to analyse all of this data. In the event the USV goes offline we can roughly determine where and why it went offline. If current and voltage levels have slowly dropped to zero we can assume the batteries have ran out of charge and the GPS that the fourth year IoT students implemented can show us the last known location of the USV and a retrieval mission can be started.

Vision Systems to radio network –

The vision systems aspect of this was created by (John) Oladipupo Shotade, what he created exactly was a ROBOrealm script that would track a green plastic ball and write serial command depending on the balls location on screen.
My implementation of this sub-project was to implement radio signals that would be send to the microbit receiver so that the thrusters on the USV would move according to the position the ball was onscreen,
This was done by simply reading the last serial command on the microbit and then sending that out onto radio group 1, the receiver would then pick up on this and activate the thrusters based on what serial command was being sent.
This allowed for near instant control of the USV by just moving a green plastic ball in front of a camera.

Current/Voltage readings (baseboard to microbit to cloud) –

The Current and voltage readings code was done by (David) Mayowa Adegbesan and Patryk Goluszka from electronic engineering year 3, They had microbit code that would calculate the voltage and current readings and print them to the serial console, My implementation was to take these values and send them out onto the microbit radio network to be accessed by the DFRobot microbit which would then send the values to BeeBotte for an online view of the current and voltages.

Implementation

In order to achieve the outcome I arrived at these are the steps I took

Learning how to use the microbit interface:

With a background in other programming languages such as Java, JavaScript, and Python learning to use Microbit wasn't that difficult.

As a start I started doing some basic "Hello World" coding on the microbit once I had these as a baseline, I then applied them on a larger scale

Here are some examples of the Hello world code I wrote and how I scaled them up:

Radio Comms: First I made two microbits send numbers and strings between each other once had that working I was confident to move onto something a bit more difficult, I decided to make a dedicated sender and receiver so I could have the receiver stationary in one place and wired up to an actuator like a servo motor or an LED and the sender could be anywhere in range of the receiver. This was pretty much the USV taking its first steps because I had a microbit connecting to a receiver microbit that was connected to a motor.

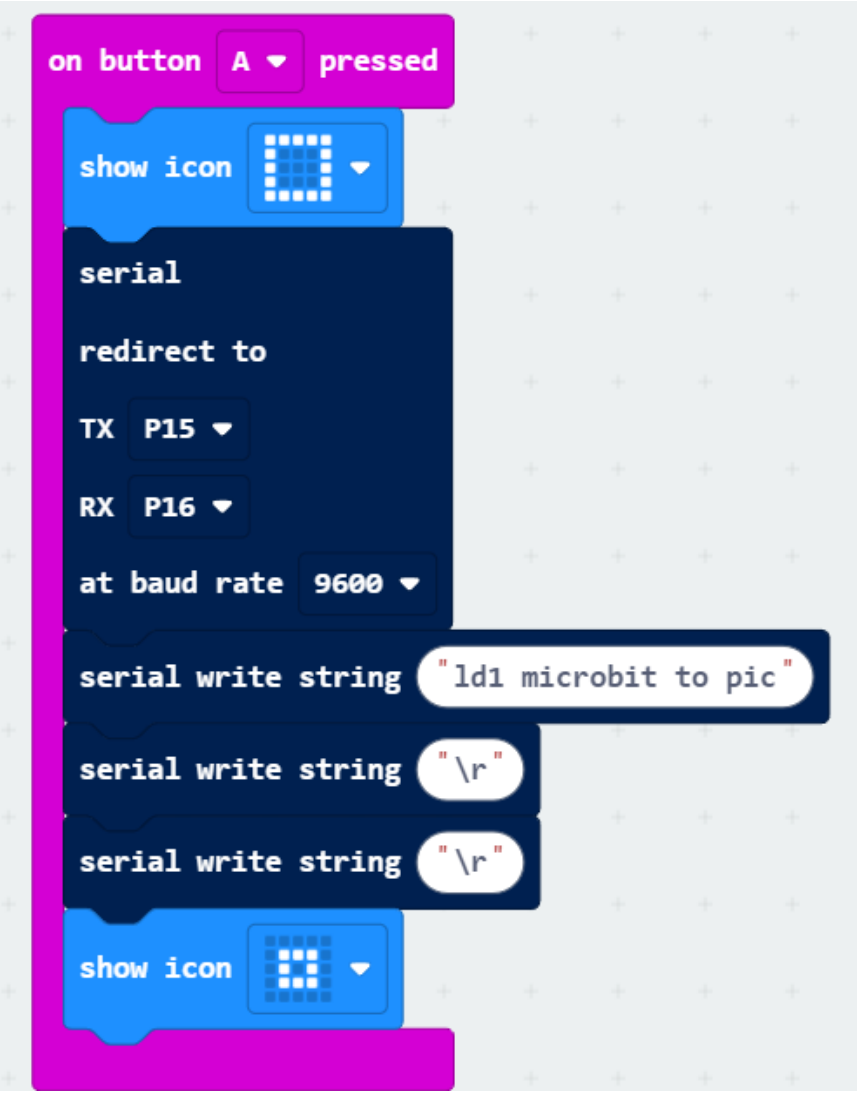
After I had radio comms working, I moved onto serial comms.
Serial coms were important to get working because this is how the microbit is able to talk with the baseboards and ROBOrealm programs.

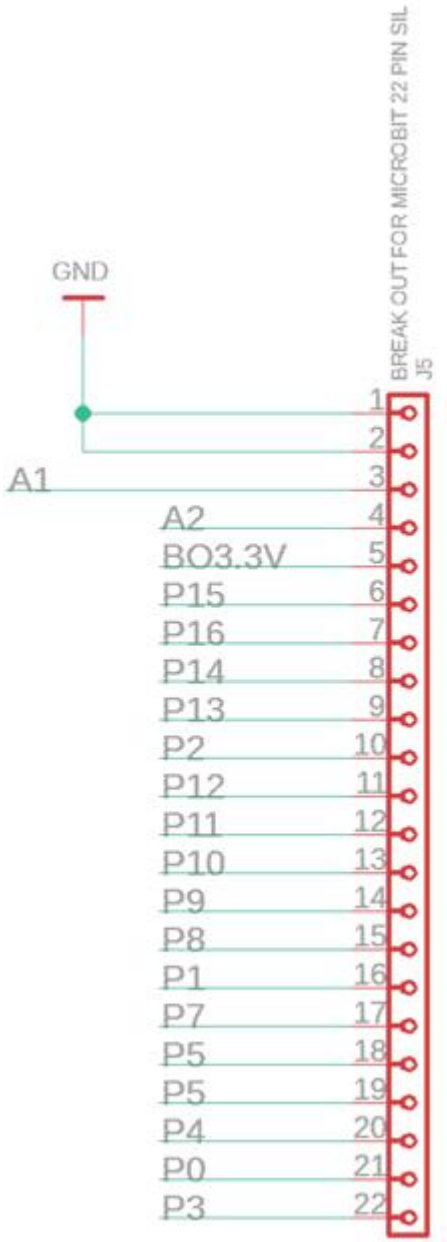
Serial Comms was very similar to radio comms in the sense that there is a sender(microbit) and receiver (serial monitor).

My first hello world for serial comms was getting a message to show up on the serial monitor using putty, while setting up putty the baud rate had to be set to 115200 because that is the speed at which the microbit communicates at.

After the hello world was completed, I moved onto getting the microbit to print to the baseboards serial monitor, this was achieved by using a serial redirect make block because we needed to change where the microbit was sending and transmitting on its pins so it would match the pins on the baseboard and using the kitronik motor driver as a "shield" for the baseboard.

Microbit to Board : https://makecode.microbit.org/_70pLuJD8jaXR





Radio Comms Hello World: https://makecode.microbit.org/_AedF8YdfuUoo

Serial Comms Hello World: https://makecode.microbit.org/_KxUHgaPKJMq4

Bluetooth Serial Hello World: https://makecode.microbit.org/_VYxXbheVJLDa

Applying the basics to the USV:

With all of the testing I had done with radio and serial comms I was now able to implement it into building the radio network for the USV.

A plan was then put in place on how the USV would move with the aid of two microbit transmitters (Joysticks) and one microbit receiver (Located on the boat itself)

The Transmitters

The joysticks used microbits built in pitch(roll) value,

The pitch(roll) was constrained to values from -90 to 90 this relates to the degrees in which the microbit can turn.

That value was then mapped to values between 60 and 120 this was to accommodate the pulse width of the motors on the USV.

Finally, that mapping was then rounded to the nearest integer, this was to stop the microbit from sending data that was too accurate because we did not need pinpoint accuracy for these joysticks.

Instructions:

- **For this code to work you must have the same radio group set on the receiver micro-bit. Then simply flash the micro-bit and tilt the micro-bit back and forth to move the motors connected to the receiver.**

Transmitter Code: https://makecode.microbit.org/_Cm90upfqvAPM

The Receiver

The receiver microbits job was to receive the name and value from the sender microbit and then apply whatever value it was given to the motor.

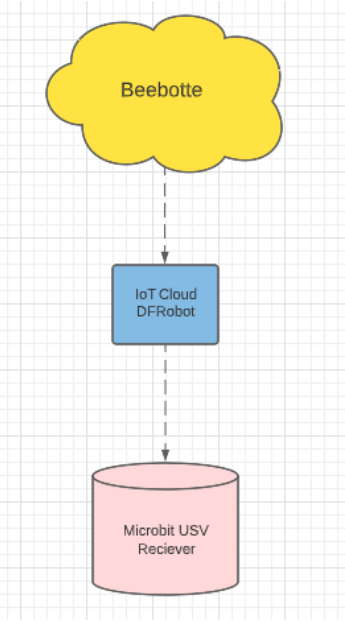
E.g. If a radio signal that was send has the name “right” and the value is 120 (after calculation),then the USV’s right motor would move at full speed.

Instructions:

- **For this code to work you must have the same radio group set on the joystick micro-bit this is because there is safety code on the receiver that will disable motors if the joystick isn’t sending radio signals.**

Reciever Code : https://makecode.microbit.org/_hLUL29907Vq4

Beebotte to Reciever:

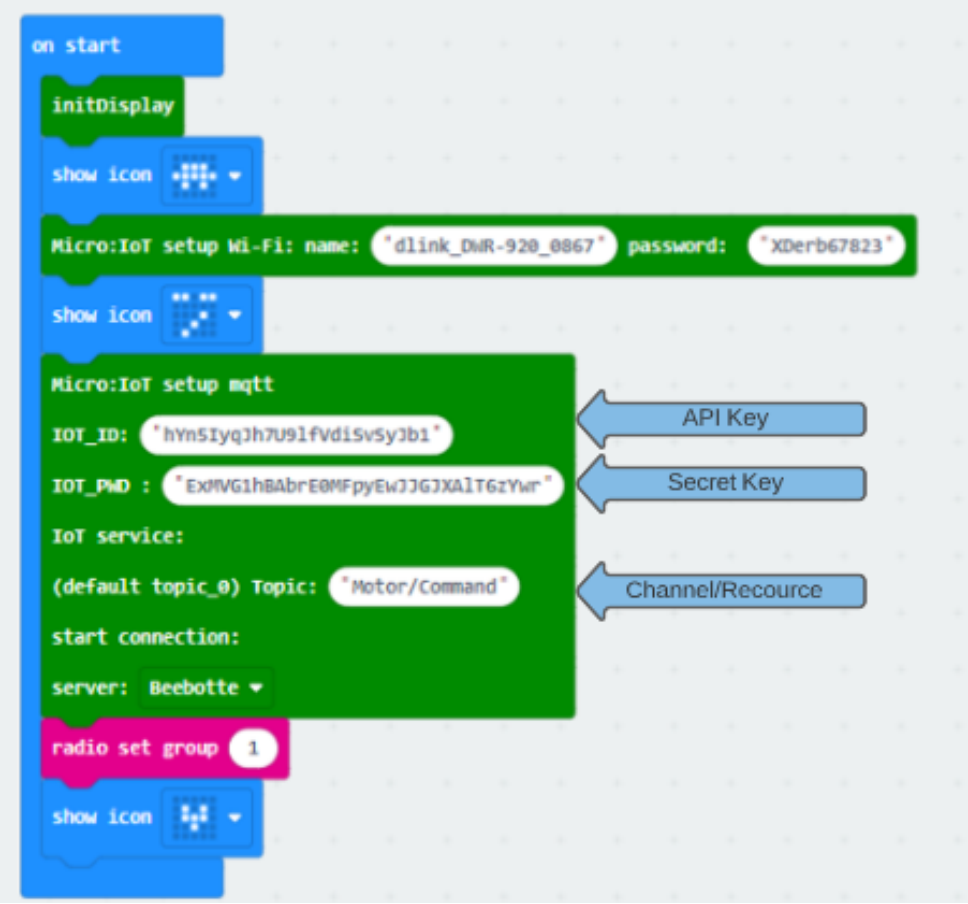


The IoT Cloud DFRobot was used to connect the USV to beebotte,

The DFRobot microbit has code that will receive messages that Are sent down from beebotte, these messages sent down from beebotte are in a JSON format, the code manipulates the JSON Files to send out the important data. An example would be “left 120” command would make the left motor move at full speed. (see pg.8 for motor speed values).

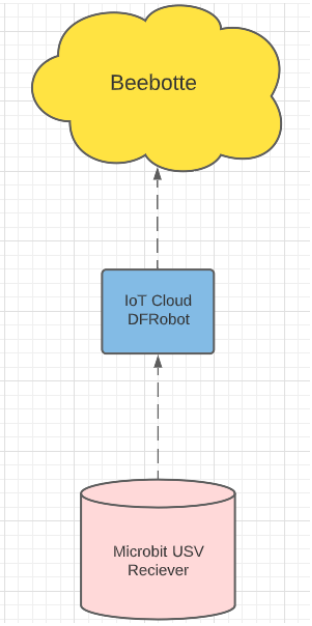
Instructions:

- **For this code to work you must have an account set up with Beebotte. The API, Secret key and Channel/Resource details must be in the following locations on the micro bit code.**



Code: https://makecode.microbit.org/_MsxYkF66pcJz

Voltages and current to Beebotte:



Once agin the IoT Cloud DFRobot was used to connect the USV Reciever to the Beebotte network.

The microbits connected to the baseboard are connected up to the batteries on the USV, the code then does some calculations to determine what voltage/current is going through the batteries at any given time. That data is then sent out on the microbit radio network for the DFRobot to receive and send to the Beebotte servers.

Baseboard to Current : https://makecode.microbit.org/_V5pUyFaoxDhp

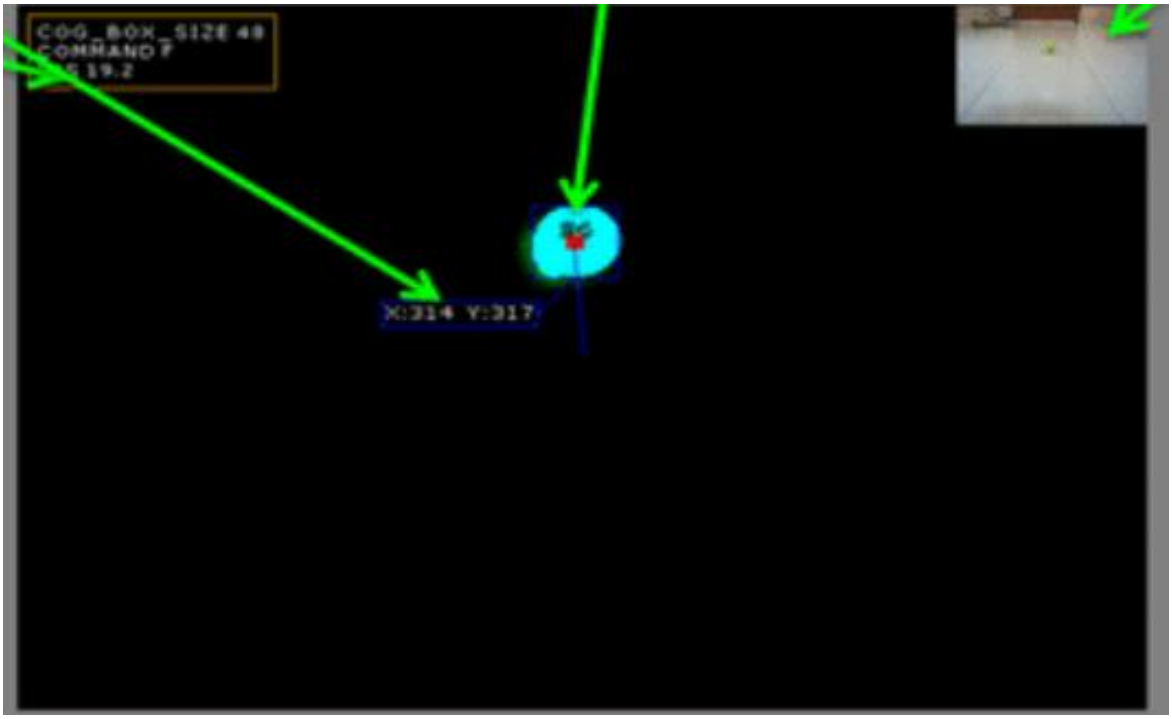
Current to Cloud : https://makecode.microbit.org/_fi67gy3TM9Eh

Autonomus movement using microbit:

ROBORealm is a robotics vision systems software that allows OpenCV functionality with very little code.

The Autonomus movement was done by creating a roborealm script that would highlight a green ball onscreen and serial print its location, for example if it was on the left hand side of the screen it would serial print “left”.

Microbits were implemented by reading the serial monitor and then sending the printed value out on the microbit radio network. This allowed the USV thrusters to move according to where the ball was onscreen.



Code: https://makecode.microbit.org/_5EiL7piWki2F

Project Conclusion & Recommendations:

Conclusion:

In Summary the project has been a success, data such as voltage and current levels of the motors were able to be sent up from the USV and down from an IoT platform (BeeBotte), this is important because this can be implemented in many projects down the line because 90% of projects consist of analog ins and analog outs this information when passed on to future students will save time and allow students to do other to work on other important features.

On a personal note I feel like I have become a much better developer from this module because being in the lab working with this system in real time and being able to make quick changes whenever things went wrong in testing has changed how I write my code forever, I now thoroughly inspect my code and try to understand why something does or does not work and if it can be simplified then simplify it to the extreme, using the micro bit interface seemed like a step backwards at first because it is a program intended to help children learn the basics of coding but I have found sometimes the most elegant solution to a problem isn't about how complicated it is but how simple you can actually make it.

Recommendations:

For future students taking on my implementation of this project I would highly recommend practicing using the radio network with a few "Hello World!" tests and once comfortable try writing a script that would make the USV move autonomously for example make a transmitter microbit so that once Button A is pressed it would execute a variety of commands to make the USV move around a body of water.

In addition to this a self-balancing feature would be nice to see this could be done by using the microbits built in compass feature so that if the USV is told to move forward it and there is a strong current it will move the motors at such speeds that it would balance itself and move only forward and not drift.