

Final Project – Bluetooth and IR

Samuel Poff, Chanartip Soonthornwan

CECS 447, M/W (10:00)

May 7, 2018

Table of Contents

[Introduction: 2](#_Toc513482778)

[Operation: 3](#_Toc513482779)

[Hardware Design: 5](#_Toc513482780)

[Top Level Schematic: 6](#_Toc513482781)

[List of Hardware Components: 6](#_Toc513482782)

[Software Design: 7](#_Toc513482783)

[Programing flow: 9](#_Toc513482784)

[Conclusion: 10](#_Toc513482785)

Table of Figures

[Figure 1: Bluetooth Controller Interface 3](https://d.docs.live.net/95182daea981e71a/2018Spring_CSULB/CECS447_Microcontroller/Projects/FinalProject/report/CECS447_%20Project3.docx#_Toc513482786)

[Figure 2 Top Level Block Diagram 6](#_Toc513482787)

[Figure 3 Full Project Picture 7](#_Toc513482788)

[Figure 4 Programming flow 9](#_Toc513482789)

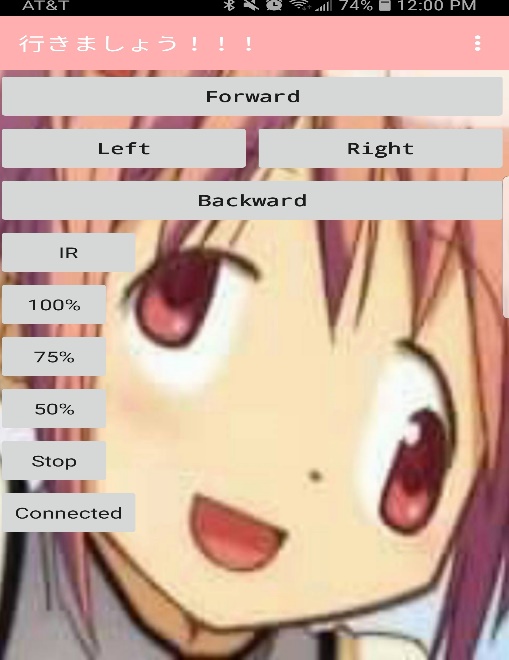
# Introduction:

In this project, there are two parts as one is a car controlled by Bluetooth controller which the car could transmit a frame of modulated IR signal as commanded by the Bluetooth controller, and another part is an IR receiver which would receive IR signal and display the result on an LCD display. Both parts are controlled by a TM4C123G on each side. The Bluetooth controller is capable of controlling the car in different directions, stop, and with different speeds. The objective is to drive the car into the range of IR receiver, send IR signal to the receiver and wait for the receiver to response.

# **Operation:**

The receiver is placed to wait for the car sending IR signal while displaying a “waiting for signal” message on its screen. If the Receiver got a valid signal, the screen would display confirmation message. Meanwhile, the car is controlled by a Bluetooth application on a smartphone to drive the car to a known destination within the range of IR transmission. The Bluetooth application consists nine buttons; W, A, S, D, Q, I, 1, 2, and 3. W is to move the car forward, A is to turn left, S is to backward, D is to turn right, Q is to stop, I is to send an IR signal, 1 is to drive the car with 100% PWM duty cycle, 2 is for 75% PWM, and 3 is for 50% PWM as shown in Figure 1. At reset, the car is stop and will drive with 50% PWM by default. To send IR signal, the car has to be in the IR receiver range confront each other to receive a valid signal.

Figure 1: Bluetooth Controller Interface



# Hardware Design:

There are two parts in this project; the car part and IR receiver part. The car’s major components are a microcontroller(TM4C123), a Bluetooth module(HC-05), a 5V regulator, an IR LED, an H-bridge module, four DC motors with four wheels, and a pack of four 3.7v lithium batteries shown in Figure 2, Figure 3, Figure 4. On the other side, the IR receiver consists of an IR receiver module, a MCU(TM4C123), and an LCD display(ST7735R) as shown in Figure 5.

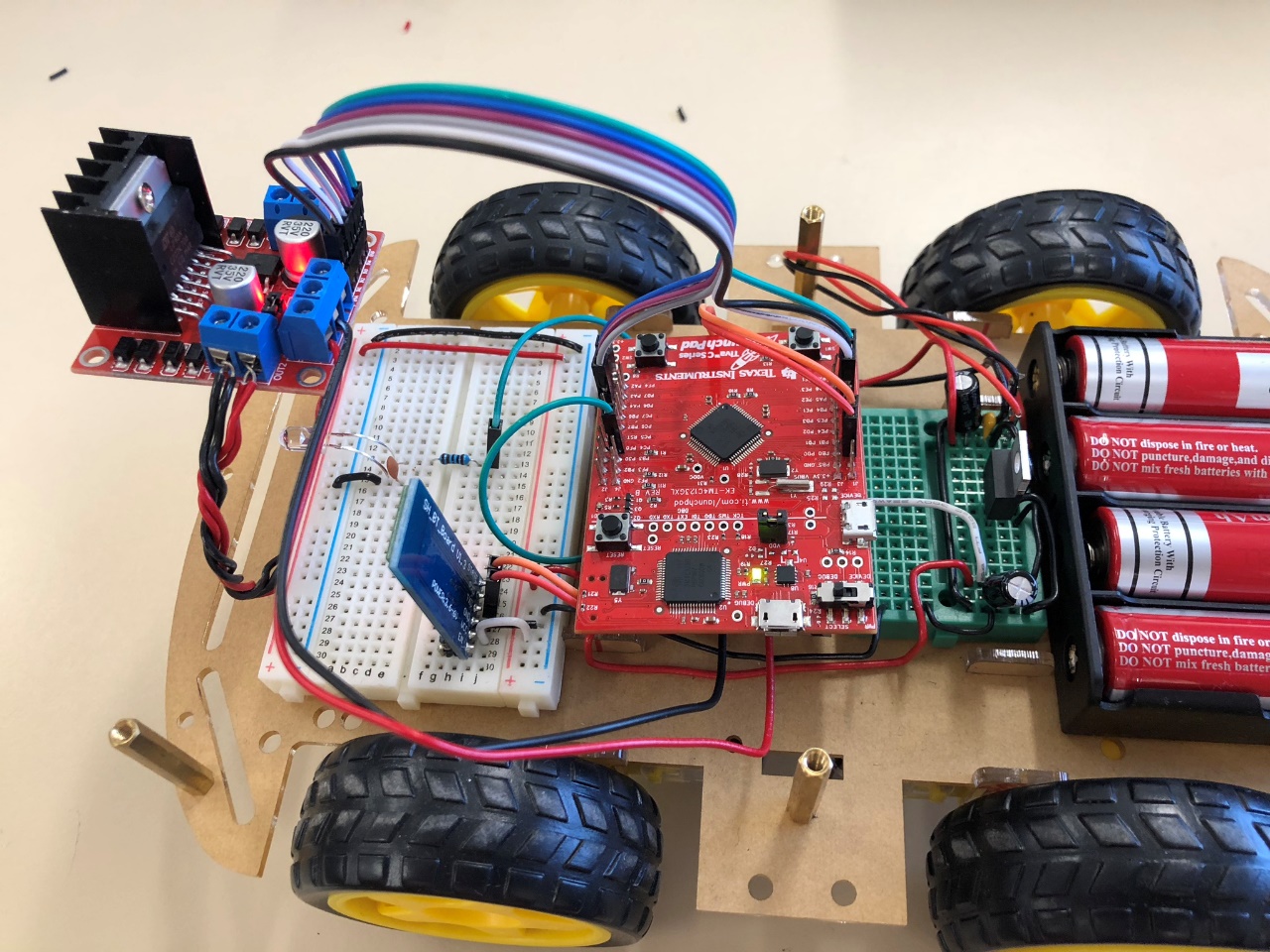


Figure 2: BT car overall connections

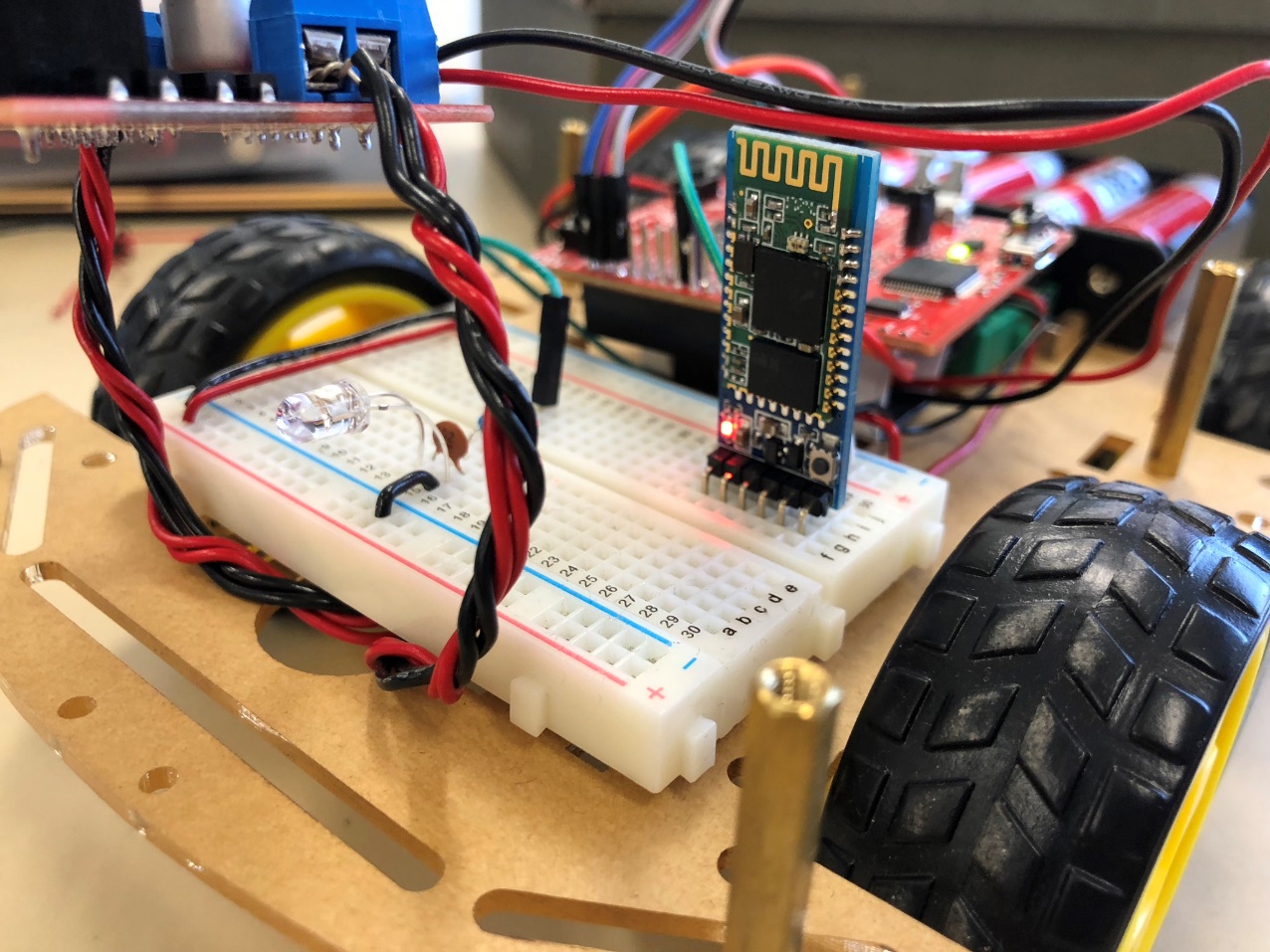


Figure 3: BT car's IR LED, BT module, and H-bridge module

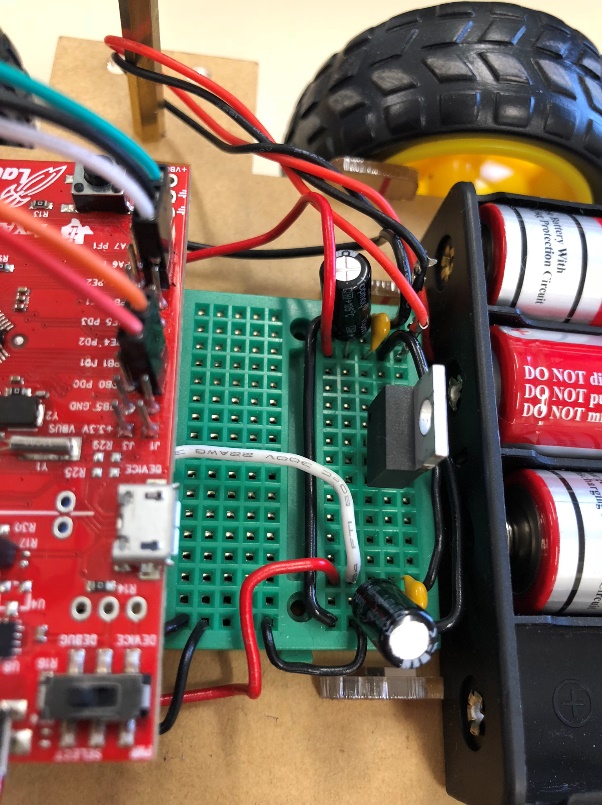


Figure 4: BT car's MCU, 5V regulator, and battery pack

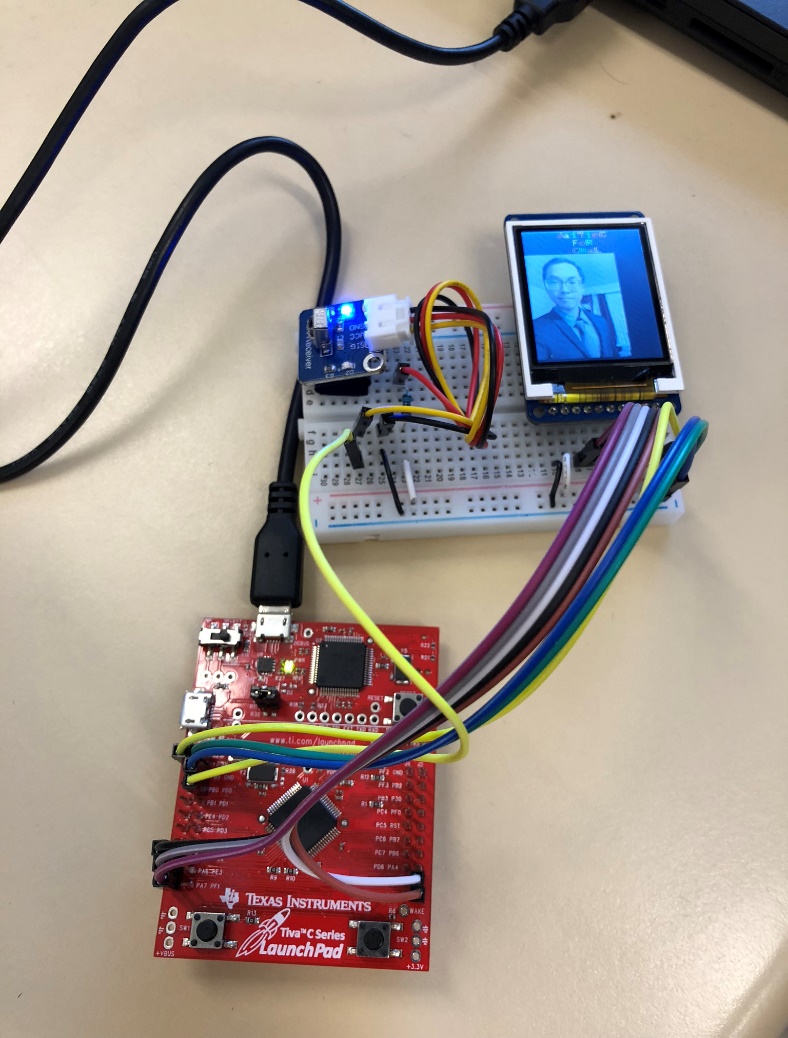


Figure 5: IR Receiver with MCU and LCD

## Top Level Schematic:

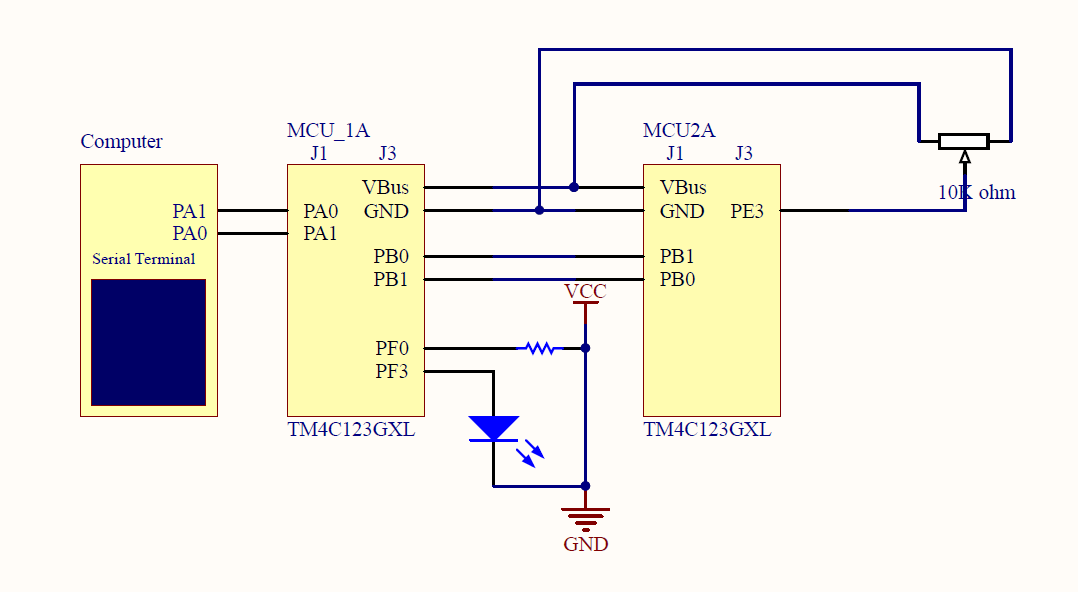


Figure 6 Top Level Block Diagram

## List of Hardware Components:

2 TM4C Lauchpad Microcontrollers

1 10K Potentiometer

2 USB Connector Wires

2 Laptops

5 Jumper Wires

1 Breadboard

**Full Project Picture:**

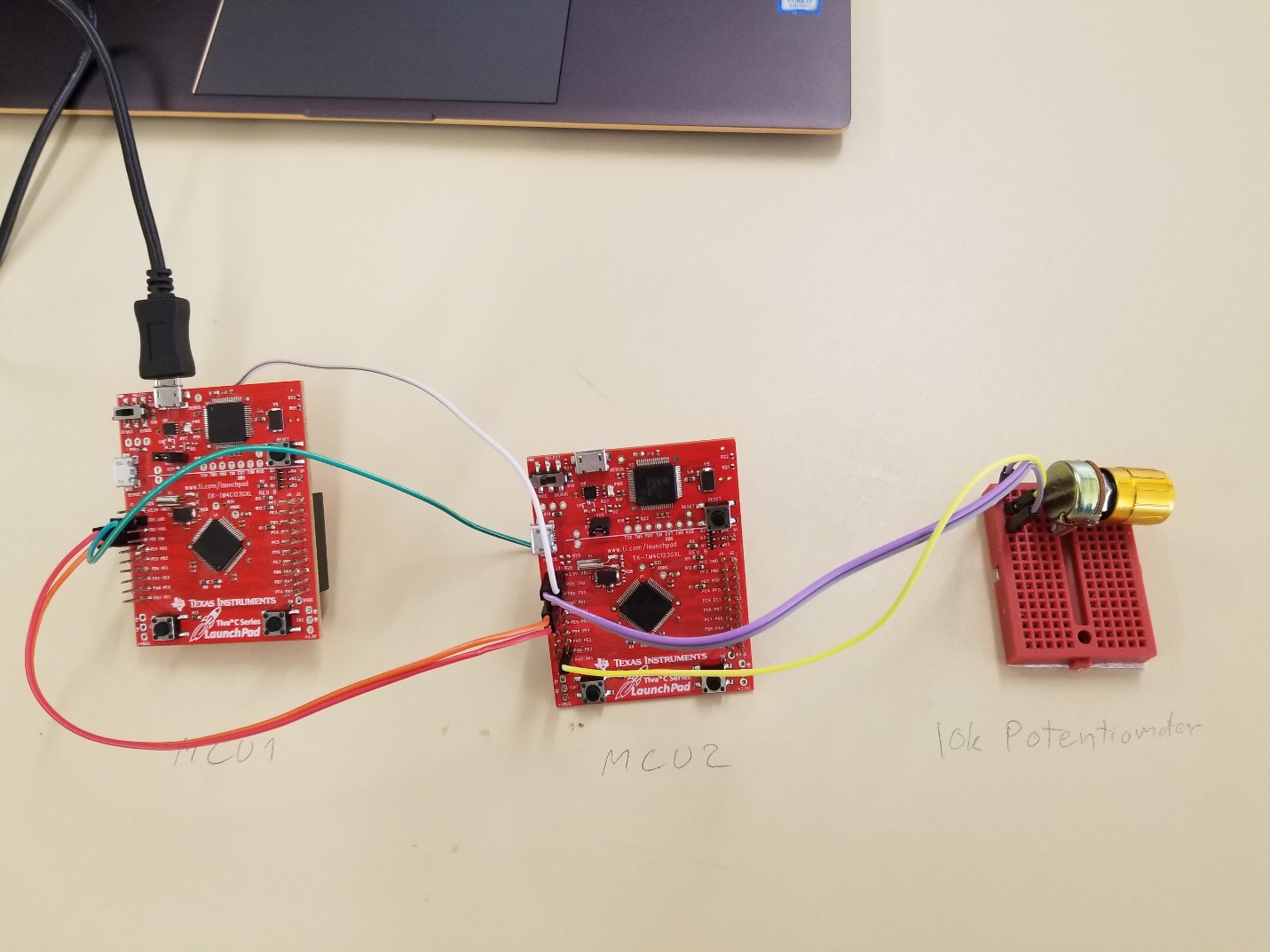


Figure 7 Full Project Picture

# Software Design:

This project’s software consisted of two programs, one for each of the two MCUs. The software for MCU1 takes care of outputting characters to the terminal, taking in input from the keyboard, and outputting a pulse modulated signal from its blue LED.

In MCU1, upon reset the PLL is initialized to 50MHz, UARTs one and zero are initialized to 9600 and 115200 baud rates respectively, the PWM module is initialized, and SysTick interrupt is initialized to trigger once every millisecond. When SysTick interrupts occur, a variable is iterated which acts like a timer. This variable is then used in the main loop to time the MCU as it looks for and waits for data. Sixty times per second the while loop iterates and UART0 reads in a char of data. If the data read is the token ‘r’ then it will pass it back to MCU2 as a confirmation. If it does not receive the token it will read in a character again. This time if it sees an ‘r’, it will attempt to read in a string of characters sent from MCU2, this will contain the message ‘Red LED is On.’ If it did not see an ‘r’ it checks to see if it saw the token ‘$’. If so, it will read in a string from MCU2 ‘Red LED is Off.’ Lastly, if that token is not received it will look for the ‘\*’ token. If it is seen, then it will read in the ADC value from MCU2 which is then passed to the PWM function to alter the blue LED’s duty cycle.

When MCU2 is reset it goes through all of the same initializations as MCU1, minus the PWM init, and adding in PortF and PortE\_ADC inits. The PortF init sets up SW0 as input and the red LED as output, and the PortE\_ADC init sets up an analog to digital converter to convert the values received from the potentiometer into a digital value. The main loop then works much like MCU1’s main loop. It reads in a character from the UART and its output is dictated by which input it sees. If it sees an ‘r’ and the button has not been pushed it will output the message ‘Red LED is On’ to MCU1. If it sees that the button has been pushed it will send the message ‘Red LED is Off’, and if it sees the ‘\*’ token it will read in the ADC value and output it to MCU1.

## Programing flow:

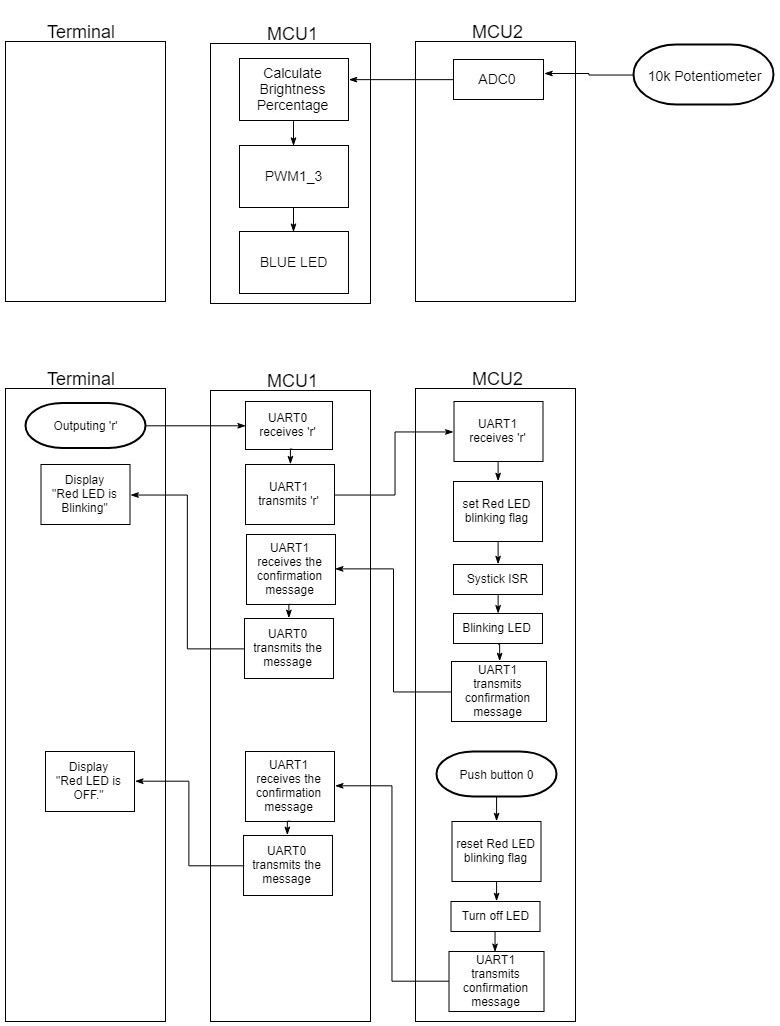


Figure 8 Programming flow

# Conclusion:

The most challenging part of this project was getting the UARTs to communicate effectively. We were able to quickly implement the PWM for the blue LED, the ADC for the potentiometer, the red LED and switch to turn it off, and even transmit the ADC values so they could be displayed in the terminal. However, we had a lot of trouble getting strings to pass between the UARTs without errors occurring in the data. The method that we attempted to use was to make each main loop iterate at roughly 60Hz in the hopes that the data would be transmitted and received close enough together that it would be coherent. If we were to attempt the project again, rather than synchronizing the main loops we would use a warning system, where the transmitting unit would send a token to inform the receiver that a message was incoming, then the receiver would hang while waiting for the data to show up.