Final Project – Bluetooth and IR

Samuel Poff, Chanartip Soonthornwan

CECS 447, M/W (10:00)

May 7, 2018

Table of Contents

Table of Figures

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

The most challenging part of this project was to transmit and receive IR signal correctly. To transmit the signal, generating 40KHz modulated signal is troublesome. Since the signal is 40KHz, we utilize 80KHz Systick Interrupt to toggle the IR LED output on each interrupting called, so a period of the signal is at 40KHz. In the meantime, while toggling the signal, we keep tracking if the signal time is in which state of the IR signal. For instance, the IR signal should toggle in Start bit high time and should stay low in low time. The problem was during the transmission the system should not have a busy wait or any approach that causes the IR signal time skew. If so the signal would not be sent properly.

On receiving side, firstly, we need to know the periods of each state on the signal and make them as references for the incoming IR signal (as shown in Figure 11). 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.

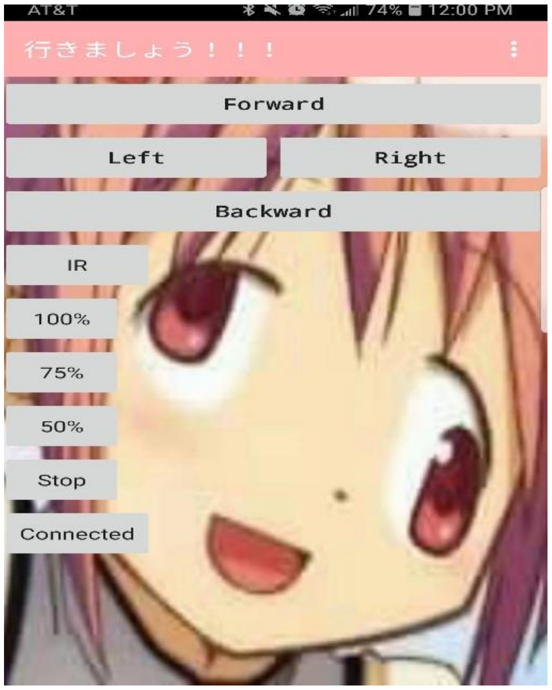


Figure 1: Bluetooth Controller Interface

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.

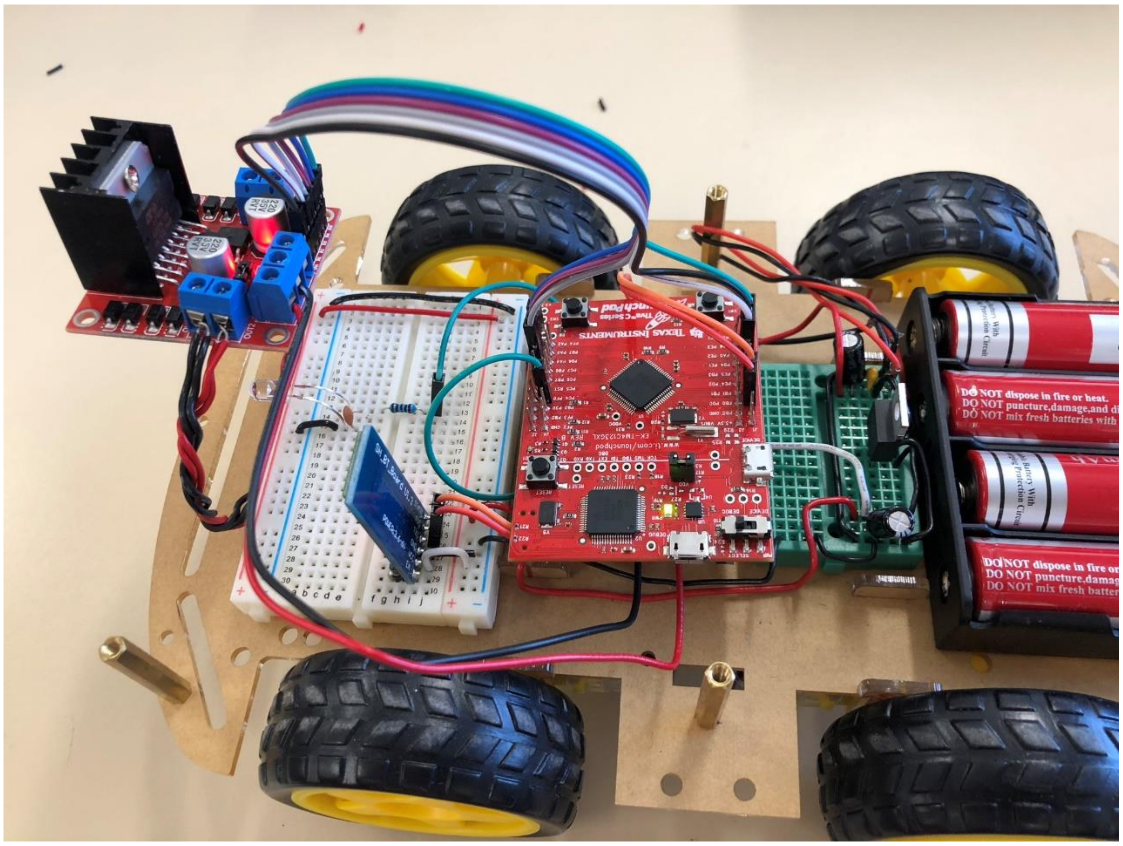


Figure 2: BT car overall connections

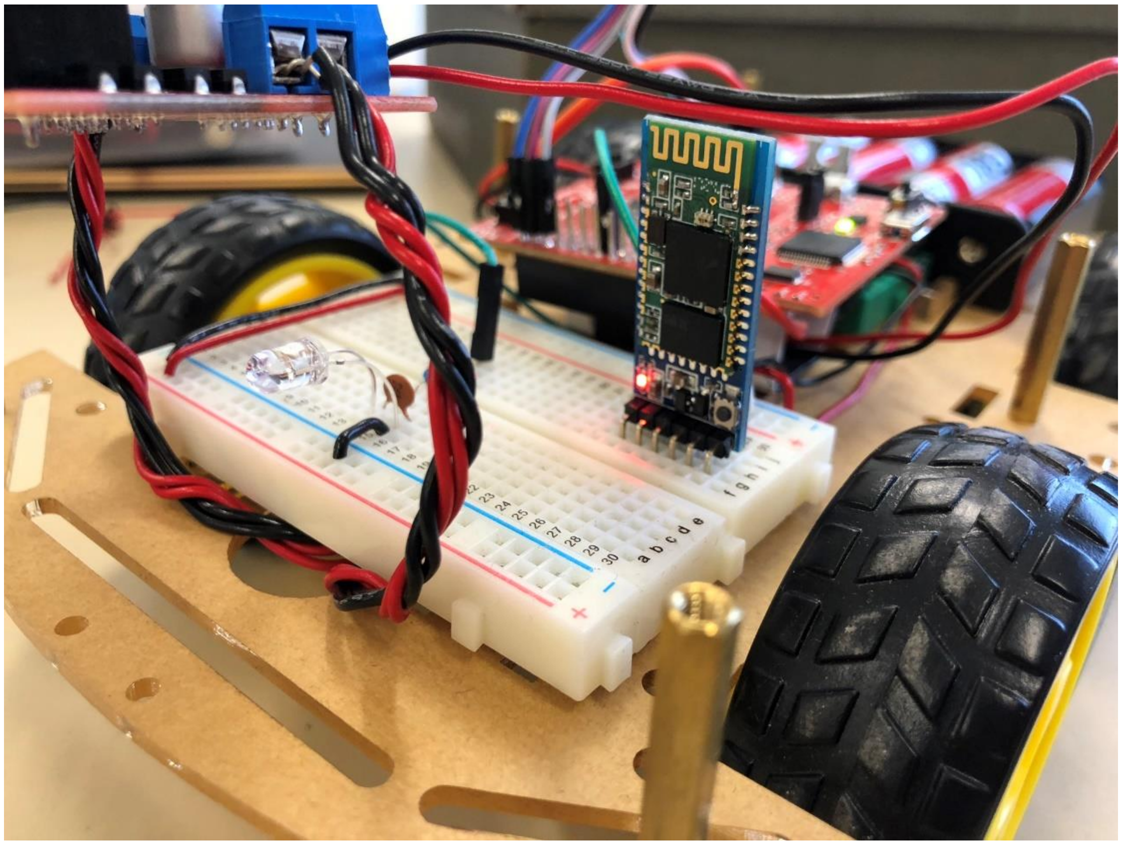


Figure 3: BT car's IR LED, BT module, and H-bridge module

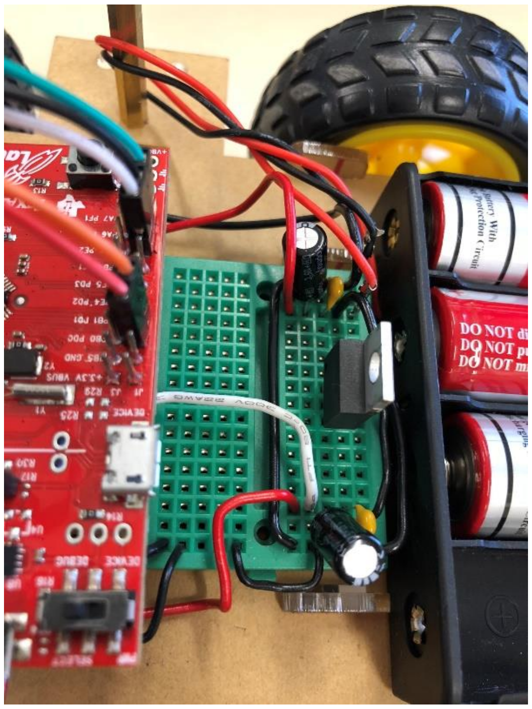


Figure 4: BT car's MCU, 5V regulator, and battery pack

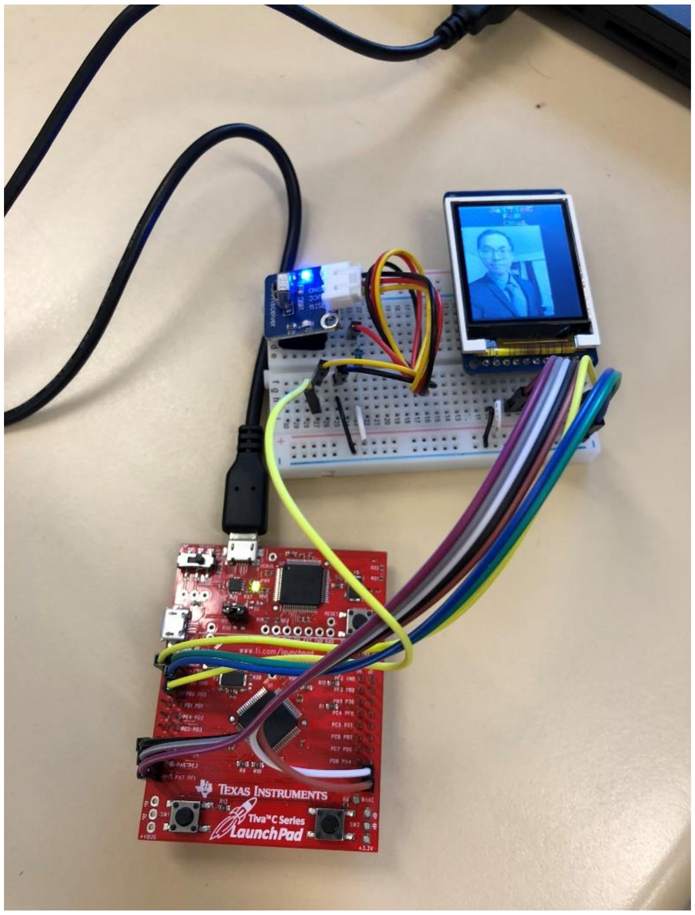


Figure 5: IR Receiver with MCU and LCD

Top Level Schematic:

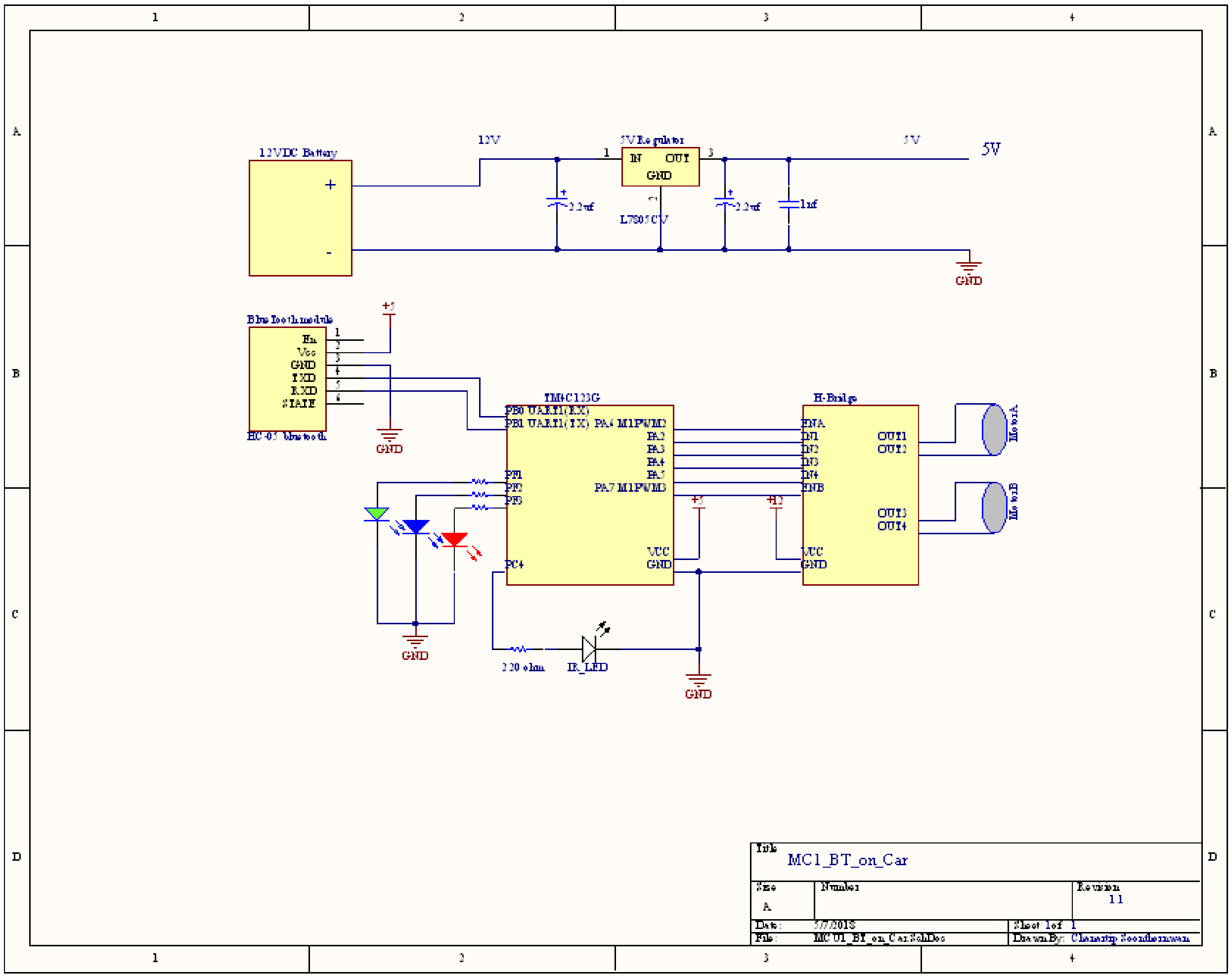


Figure 6: BT Car Schematic

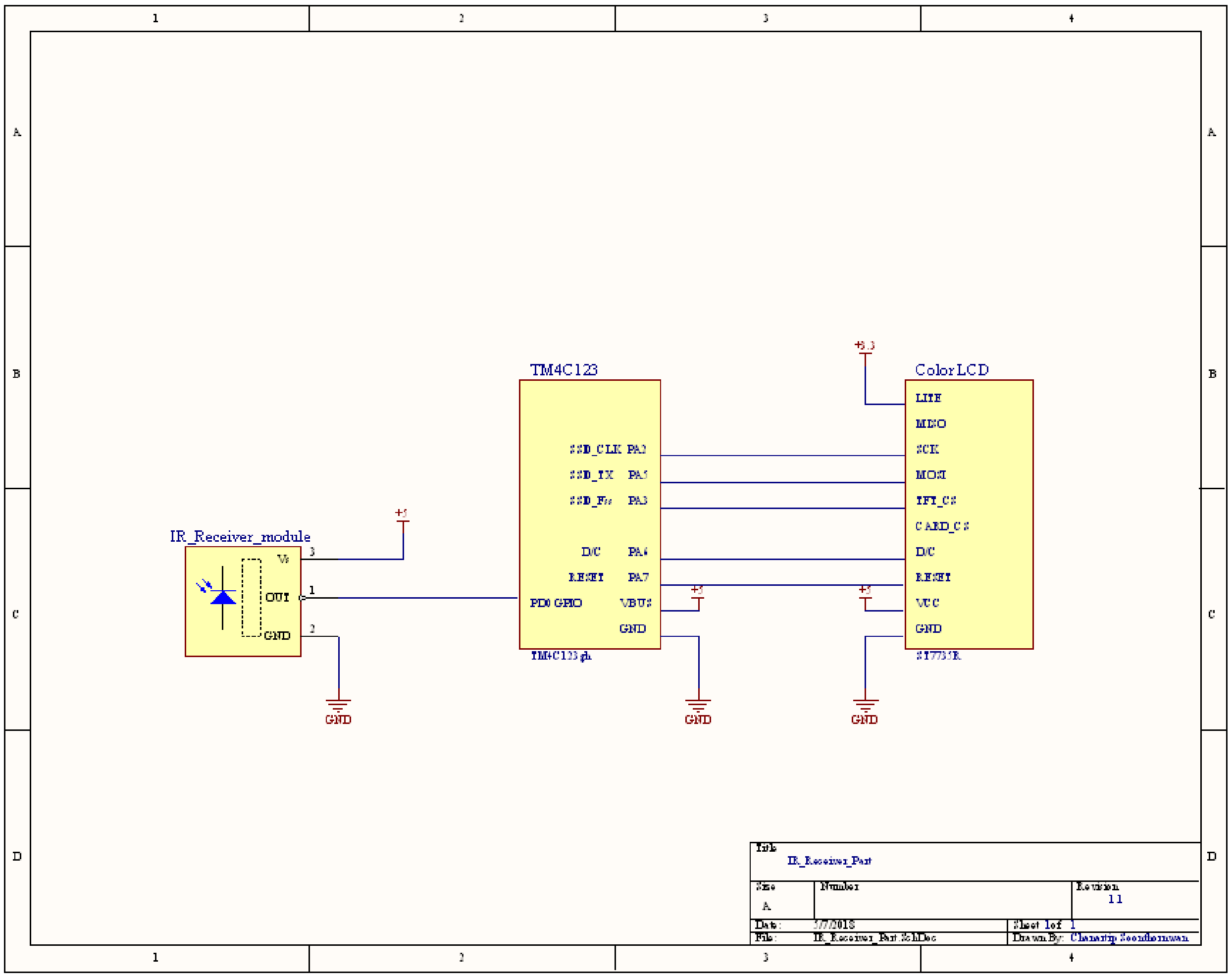


Figure 7: IR Receiver part Schematic

List of Major Hardware Components:

2 TM4C123 Lauchpad Microcontrollers

1 HC-05 Bluetooth Module

1 L7805 5V regulator

4 DC motors with wheels

1 IR LED 940nm

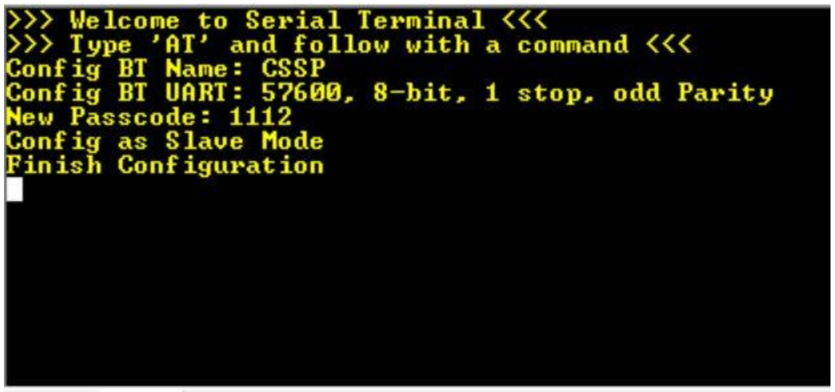
1 IR Receiver module

1 ST7735R LCD

Software Design:

As mention above, there are two parts for this project; one is the Bluetooth controlled car, and another is the IR receiver with LCD. The car itself consists three major designs, Bluetooth, DC motor control, and IR transmission.

Bluetooth control is basically transmitting and receiving data between a master (a smartphone) and a slave (the Bluetooth module HC-05 on TM4C123). Firstly, the slave is configured via consistently sending AT commands to define the slave as Name: CSSP, 57600 Baud Rate, 8-bit data, 1 stop, odd parity bit checking, and set the HC-05 in Slave Mode as shown the confirmation of the Bluetooth Configuration below. The slave will receive a character long command from its master



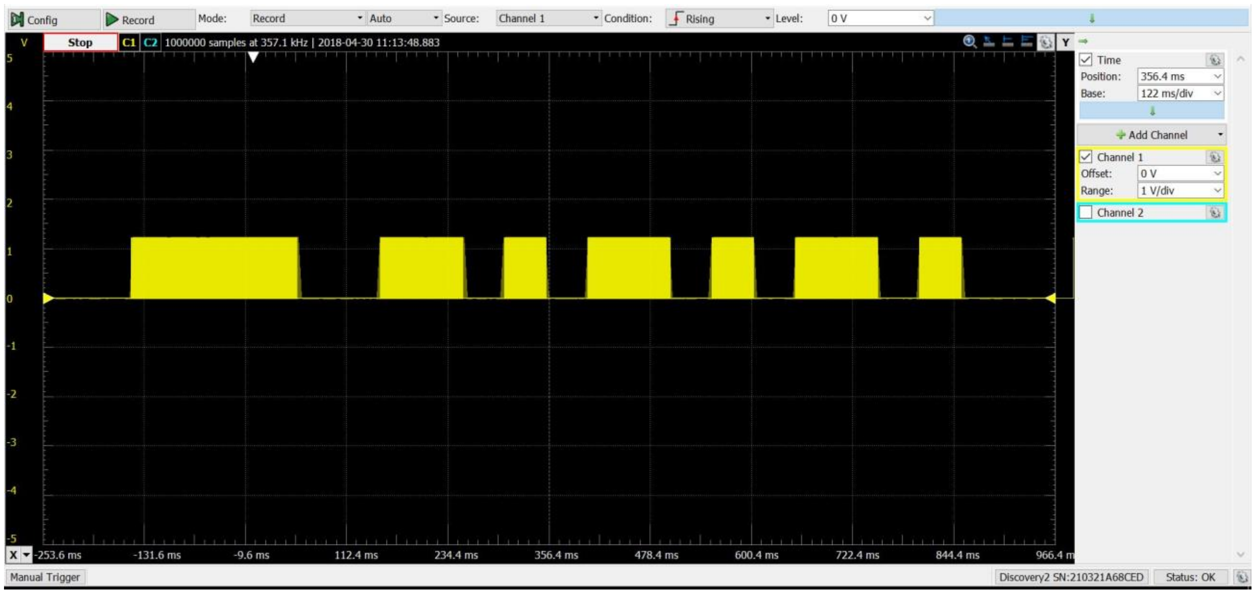
```
>>> Welcome to Serial Terminal <<<
>>> Type 'AT' and follow with a command <<<
Config BT Name: CSSP
Config BT UART: 57600, 8-bit, 1 stop, odd Parity
New Passcode: 1112
Config as Slave Mode
Finish Configuration
```

Figure 8: HC-05 configuration display on Serial Terminal.

and the program will decode the function of the command. For example, if the slave got ‘W’, the program would assign the current PWM duty to the DC motors with the set of the wheels direction of moving forward, or if the slave got ‘I’, the program would generate two frames of modulated IR signal through the IR LED.

DC-motor Control takes an action when the Bluetooth gets command for directions or speeds control. If the Bluetooth get W,A,S,D, or Q, DC motors will be assigned with current speed with a specific set of the direction of each command (as mention in Operation:). If the Bluetooth gets 1,2, or 3, the DC motors will be assigned a new PWM duty.

IR Transmission is to transmit IR signal by utilizing the Systick Interrupt to generate 40KHz modulated output through an IR LED. Once the Bluetooth gets ‘I’, it trigs the System for not continuously receiving any incoming command until the IR transmission is complete by setting a flag called IR_busy. While IR_busy is set, there is an IR counter to keep tracking in time when the modulated signal should be the Start bit, Logic 1, or Logic 0 (shown in Figure 9), while toggling the signal in its high time (shown in Figure 10).Once the IR transmission finishes, the program release IR_busy, so the system could receive new command.


Mode: Record | Auto | Source: Channel 1 | Condition: Rising | Level: 0 V

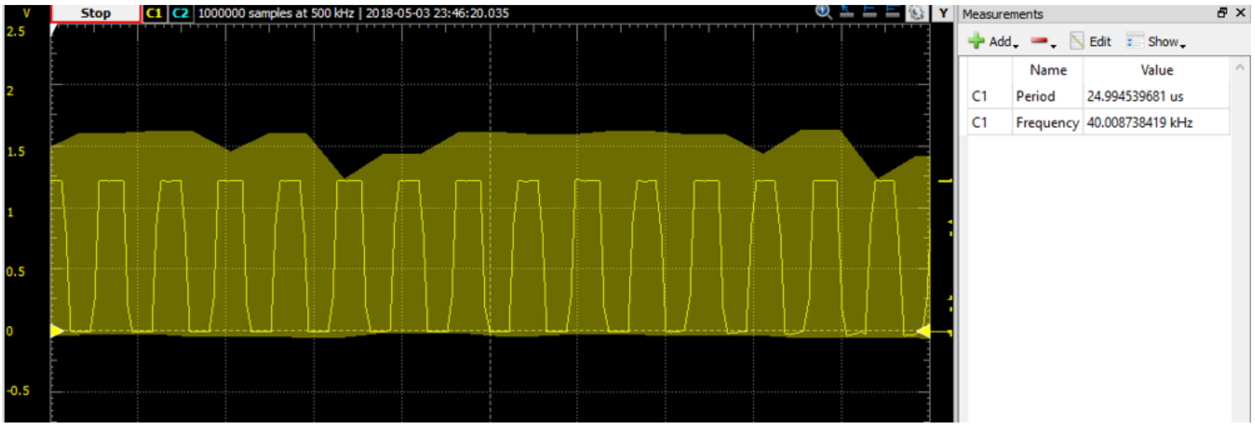
1000000 samples at 357.1 kHz | 2018-04-30 11:13:48.883

Time Position: 356.4 ms | Base: 122 ms/div

Channel 1 Offset: 0 V | Range: 1 V/div

Discovery2 SN:210321A68CED | Status: OK

Figure 9: Modulated signal

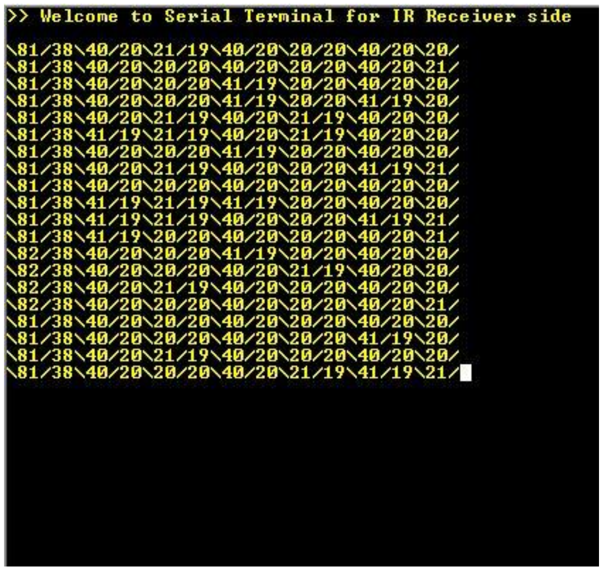


	Name	Value
C1	Period	24.994539681 us
C1	Frequency	40.008738419 kHz

Figure 10: IR signal in High Time

On the other side, IR receiver with LCD utilize Systick Interrupt to consistently checking the input from IR-receiver module if there is a falling edge input occur stating that it probably is the start bit. Then, the interrupt sets a flag called Got_IR, reset IR timer, and number of the signal sample, therefore, the interrupt could only sample the input signals if they are valid in each stage's period. If the inputs are not valid in the period, an error counter is incremented for later calculation. When the IR timer is reached 420 ticks(10.5ms) the interrupt would release the Got_IR and calculate the Error percentage. If the Error percentage is higher than 10 percent, the input is invalid, and the LCD will display the "Waiting for Signal". Otherwise, the LCD will display "Got Signal" to confirm that the transmission is complete.

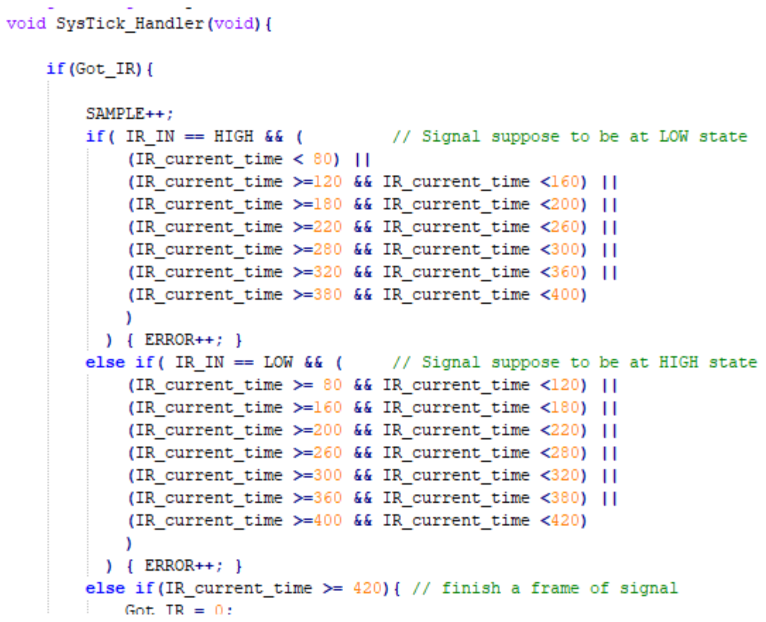
To find the periods of the signal, we create a code to check if the receiver gets start bit, and then displays the IR periods on each time the inputs do falling edge or rising edge (as shown in Figure 11) and utilizes the period to check error of the actual demodulated signal since both the code periods and actual periods should be similar.



```
>> Welcome to Serial Terminal for IR Receiver side

\81/38\40/20\21/19\40/20\20/20\40/20\20/
\81/38\40/20\20/20\40/20\20/20\40/20\21/
\81/38\40/20\20/20\41/19\20/20\40/20\20/
\81/38\40/20\20/20\41/19\20/20\41/19\20/
\81/38\40/20\21/19\40/20\21/19\40/20\20/
\81/38\41/19\21/19\40/20\21/19\40/20\20/
\81/38\40/20\20/20\41/19\20/20\40/20\20/
\81/38\40/20\21/19\40/20\20/20\41/19\21/
\81/38\40/20\20/20\40/20\20/20\40/20\20/
\81/38\41/19\21/19\41/19\20/20\40/20\20/
\81/38\41/19\21/19\40/20\20/20\41/19\21/
\81/38\41/19\20/20\40/20\20/20\40/20\21/
\82/38\40/20\20/20\41/19\20/20\40/20\20/
\82/38\40/20\20/20\40/20\21/19\40/20\20/
\82/38\40/20\21/19\40/20\20/20\40/20\20/
\82/38\40/20\20/20\40/20\20/20\40/20\21/
\81/38\40/20\20/20\40/20\20/20\40/20\20/
\81/38\40/20\20/20\40/20\20/20\41/19\20/
\81/38\40/20\21/19\40/20\20/20\40/20\20/
\81/38\40/20\20/20\40/20\21/19\41/19\21/
```

Figure 11: Displaying periods of input level changes



```
void SysTick_Handler(void){

    if(Got_IR){

        SAMPLE++;
        if( IR_IN == HIGH && (          // Signal suppose to be at LOW state
            (IR_current_time < 80) ||
            (IR_current_time >=120 && IR_current_time <160) ||
            (IR_current_time >=180 && IR_current_time <200) ||
            (IR_current_time >=220 && IR_current_time <260) ||
            (IR_current_time >=280 && IR_current_time <300) ||
            (IR_current_time >=320 && IR_current_time <360) ||
            (IR_current_time >=380 && IR_current_time <400)
            )
          ) { ERROR++; }
        else if( IR_IN == LOW && (      // Signal suppose to be at HIGH state
            (IR_current_time >= 80 && IR_current_time <120) ||
            (IR_current_time >=160 && IR_current_time <180) ||
            (IR_current_time >=200 && IR_current_time <220) ||
            (IR_current_time >=260 && IR_current_time <280) ||
            (IR_current_time >=300 && IR_current_time <320) ||
            (IR_current_time >=360 && IR_current_time <380) ||
            (IR_current_time >=400 && IR_current_time <420)
            )
          ) { ERROR++; }
        else if(IR_current_time >= 420){ // finish a frame of signal
            Got_IR = 0;
```

Figure 12: Applied tested period in Systick Interrupt

In order to confirm the signal visually, we use Analog Discovery2 to scope the output of modulated IR signal from the IR LED on the car and demodulated IR signal from the IR Receiver module, and the display both signals on top of each other with slightly shifted as the signal takes time traveling in the air (shown in Figure 13).

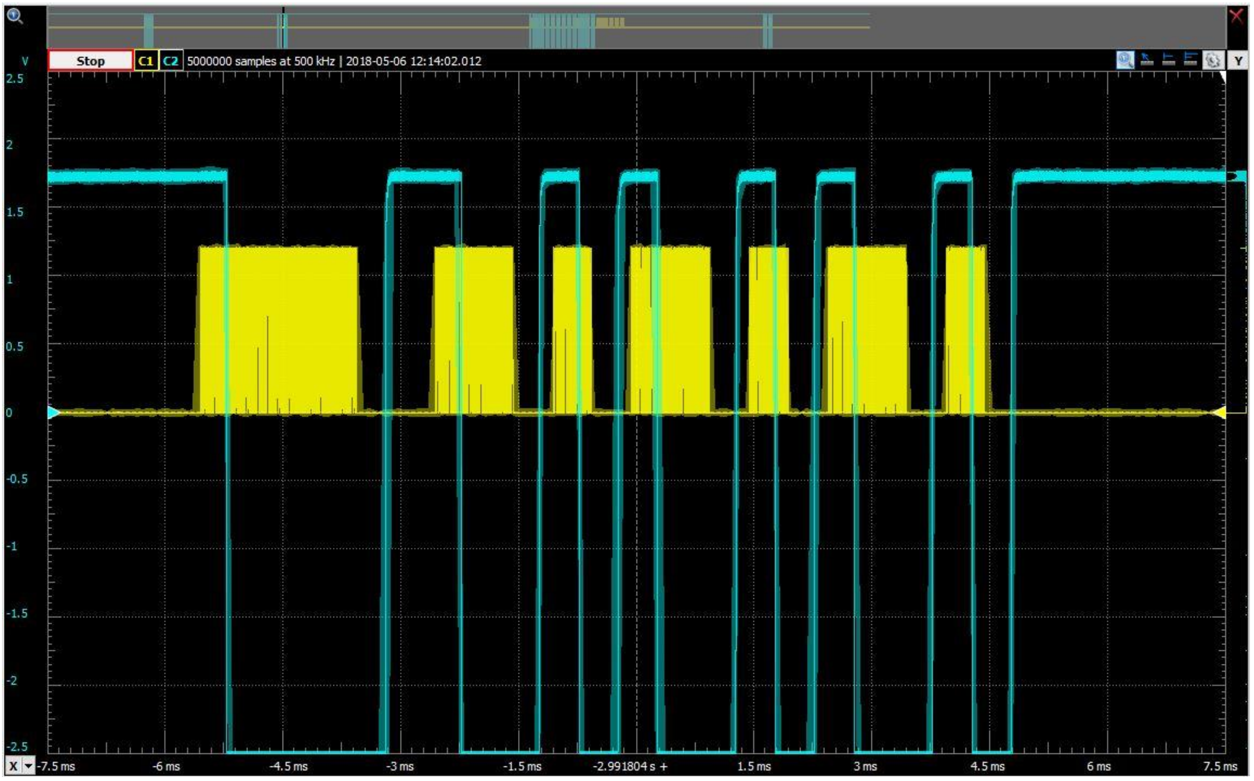
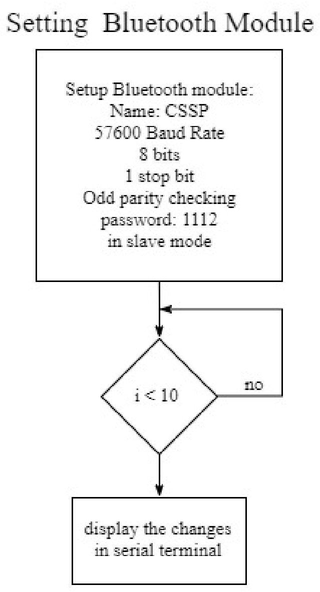
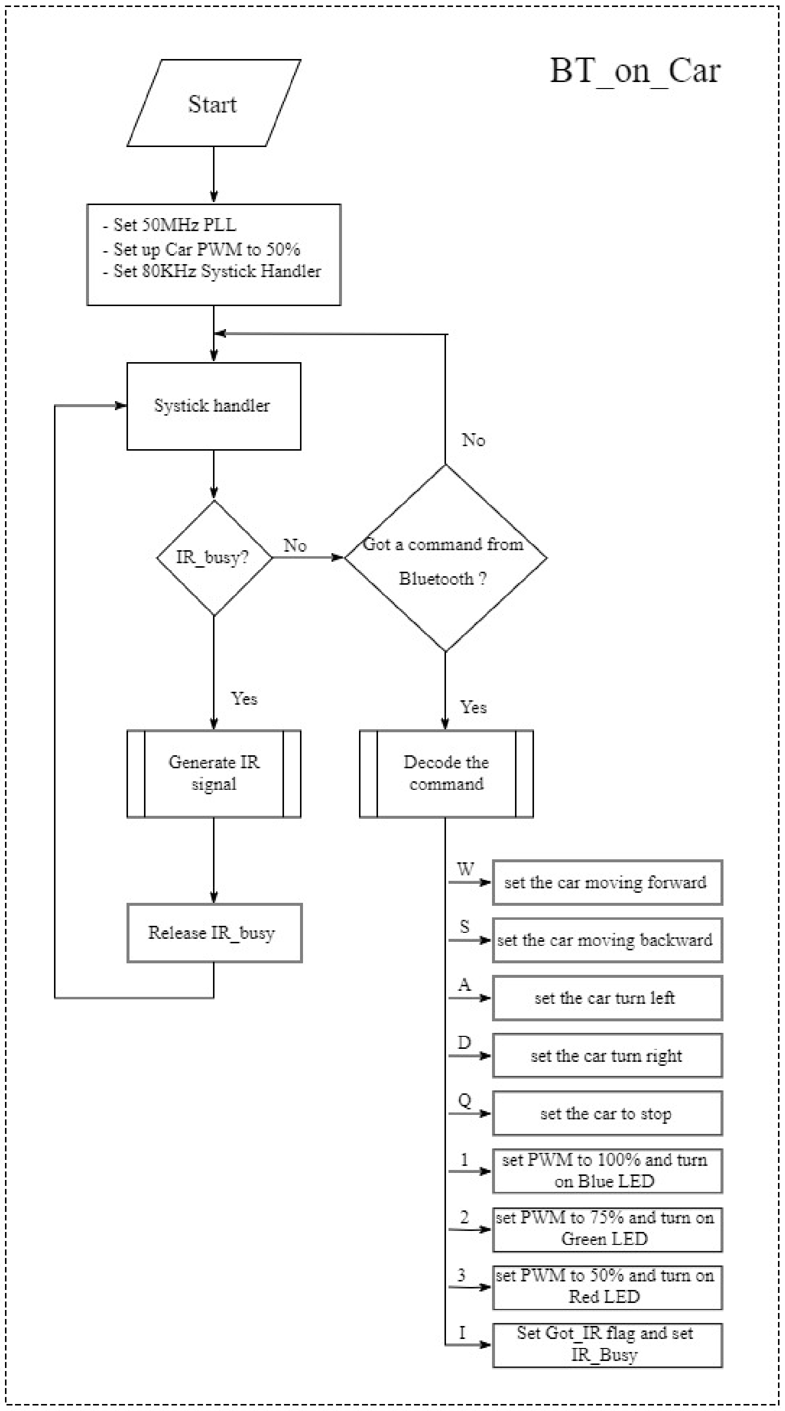


Figure 13: Modulated and Demodulated IR signal in Yellow Cyan

Programing flow:



Setting Bluetooth Module
Setup Bluetooth module: Name: CSSP 57600 Baud Rate 8 bits 1 stop bit Odd parity checking password: 1112 in slave mode
i < 10
no
display the changes in serial terminal



BT_on_Car
Start
- Set 50MHz PLL
- Set up Car PWM to 50%
- Set 80KHz Systick Handler
Systick handler
IR_busy?
No
Yes
Got a command from Bluetooth ?
No
Yes
Generate IR signal
Release IR_busy
Decode the command
W
set the car moving forward
S
set the car moving backward
A
set the car turn left
D
set the car turn right
Q
set the car to stop
1
set PWM to 100% and turn on Blue LED
2
set PWM to 75% and turn on Green LED
3
set PWM to 50% and turn on Red LED
I
Set Got_IR flag and set IR_Busy

Figure 14: BT_on_car programming flow

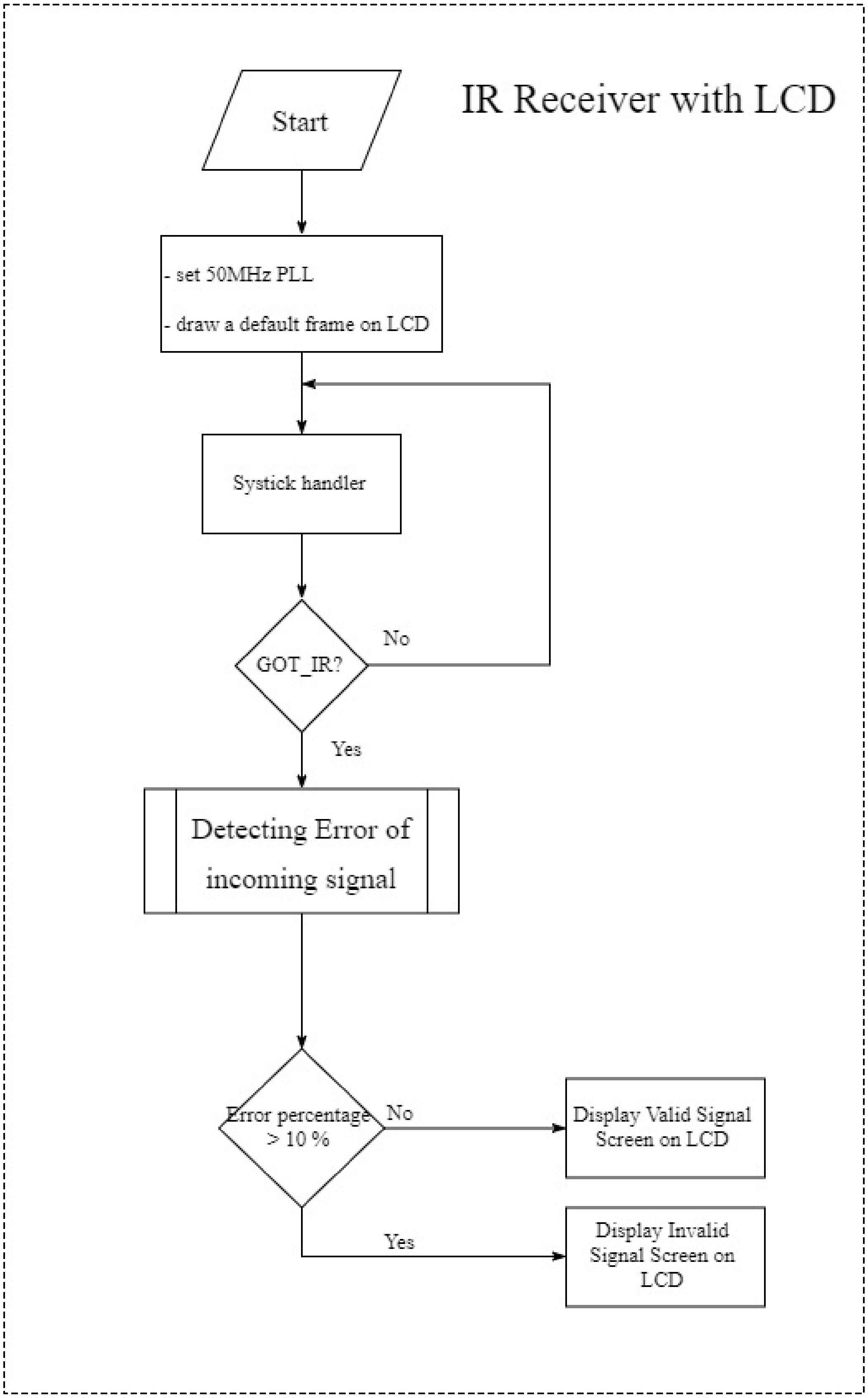

IR Receiver with LCD
Start
- set 50MHz PLL
- draw a default frame on LCD
Systick handler
GOT_IR?
No
Yes
Detecting Error of incoming signal
Error percentage > 10 %
No
Display Valid Signal Screen on LCD
Yes
Display Invalid Signal Screen on LCD

Figure 15: IR_Receiver_with_LCD programming flow

Conclusion:

The most challenging part of this project was to transmit and receive IR signal correctly. To transmit the signal, generating 40KHz modulated signal is troublesome. Since the signal is 40KHz, we utilize 80KHz Systick Interrupt to toggle the IR LED output on each interrupting called, so a period of the signal is at 40KHz. In the meantime, while toggling the signal, we keep tracking if the signal time is in which state of the IR signal. For instance, the IR signal should toggle in Start bit high time and should stay low in low time. The problem was during the transmission the system should not have a busy wait or any approach that causes the IR signal time skew. If so the signal would not be sent properly.

On receiving side, firstly, we need to know the periods of each state on the signal and make them as references for the incoming IR signal (as shown in Figure 11), so we hardcoded the period and check errors as we only have one IR command.

Another challenge is to control the car via the Bluetooth application on a smartphone. We firstly tried to press and hold a command button and then let go of the button. We found that the car was consistently moving forward as we assigned ‘W’, which made a difference from any Remote-Control car. Fortunately, the Bluetooth API allows us to setup an event when releasing a button after holding it, so we send ‘Q’ to stop the car as we release our finger off the button, which provides the better experience controlling the car. Furthermore, it could have been better if our car could actually turn left in the sense that a user holding ‘W’ and ‘A’ at the same time.

Lastly, the LCD refresh rate could have been smoother as it refreshes a new frame on the screen even though 140Hz has been used as the refresh rate for the screen, but we believe that it is the limit of the hardware itself.