CALIFORNIA STATE UNIVERSITY

1949

LONG BEACH

CECS 447 Spring 2021 Project 3

Bluetooth Controlled Robot Car

By

Jesus Franco

3/24/2021

This project will show the configuration of the Bluetooth (BT) module on the HC-05 through a serial terminal to allow BT communication between 2 BT devices to command a BT controlled robot car.

Introduction:

Part 1:

For part 1 of project 3, the purpose is to write a code that will allow the user to manually configure the HC-05 BT module to the desired qualifications for the project. This portion of it was entirely created with an Arduino to a much simpler implementation of the configuration.

Part 2:

For part 2 of project 3, the purpose was to use the newly configured BT module from the HC-05 to allow BT communication from a BT terminal to the HC-05 to control commands for the robot car. The commands will control the speed of the wheels of the robot car. This will allow the user to manually control the speed, direction, and movement of the robot car.

Port Table:

Name	I/0	Port	Description
UART1 Rx	Input	PC4	MCU2 receiving bit.
UART1 Tx	Output	PC5	MCU2 transmitting bit.
L298N ENB	Output	PB7	PWM B enable.
L298N ENA	Output	PB6	PWM A enable.
DC Motor	Output	PB3	L298N IN4.
DC Motor	Output	PB2	L298N IN3.
DC Motor	Output	PB1	L298N IN2.
DC Motor	Output	PB0	L298N IN1.

LED Logic Table:

Color	Value	Description
Green	0x08	Forward Direction.
Blue	0x04	Reverse Direction.
Yellow	0x0A	Left Turn.
Sky Blue	0x0C	Right Turn.
Blank	0x00	Stop.
Pink	0x06	Speed Up.
RED	0x02	Slow Down.

NOTE: Pink should have been the Right Turn LED instead of Sky Blue.

Operation for BT Configuration:

This process was entirely made by using the Arduino Library to configure the BT module for the HC-05. The Arduino is a much easier method to configure the BT module since it has its own serial terminal to allow a quick setup for serial communication. The Arduino code will use 2 pins to determine RX and TX. There will also be a pin dedicated for the enable pin for the HC-05 and will be set as an output and logic state HIGH. The terminal will be set to 9600 and both the RX and TX pins will be set at a baud rate of 38400 for command mode. The whole while loop of Arduino code will continue to keep reading from the HC-05 and send to the Arduino serial terminal until serial data is available so it can write and read from the HC-05 to the Arduino serial terminal. Then it will keep reading from the Arduino serial terminal and send to the HC-05 until serial data is available so it can write and read from the Arduino serial terminal to the HC-05 and keeps cycling.

Operation for BT Controlled Robot Car:

Since the heavy lifting has already been done with the BT configuration from part 1, the robot car was very simple to implement. Since the robot car circuit has already been created from the previous class, the only thing left to do is remove the extra unnecessary components from the robot car and add the BT module. There was an issue where the BT module will not be able to work since it uses PB1-0 for TX and RX and the PWM uses PB3-0 for the DC motor inputs, but it just required a tad of tweaking to allow it to work. There was the possibility of changing the port B input pins from the DC motors to other pins; however, it seemed a lot easier to just change the UART pins. I could have changed to a different UART but I found it much more easier to just change the UART1 pins to a different UART1 pins such as PC5-4. After this implementation, the rest of the code to change speed, direction, and movement was quite simple since it is very similar to the logic from the previous class.

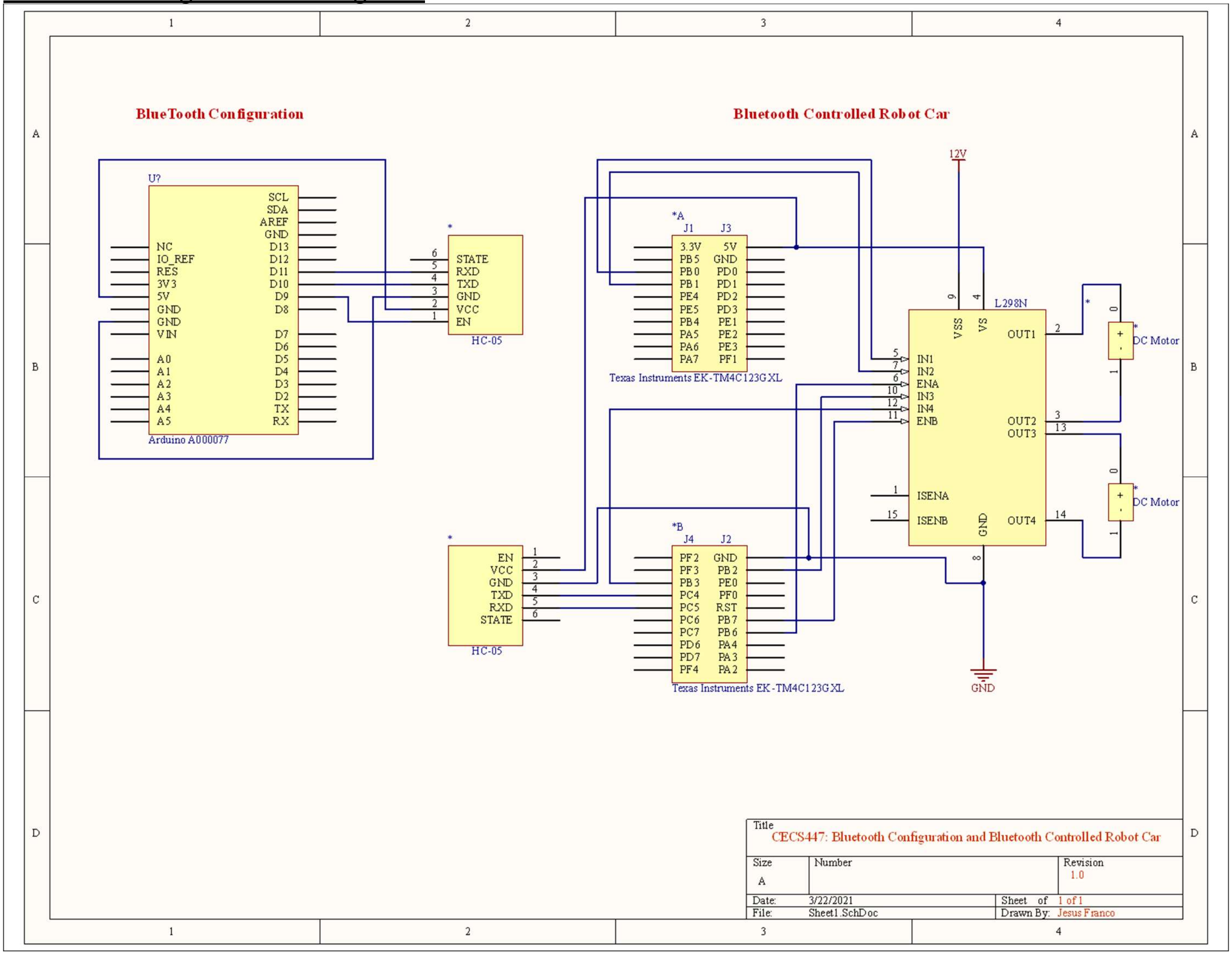
Theory for BT Configuration:

In theory, the BT configuration portion should allow the user to manually change the setup of the default BT device to the user's preferable setup options. The user will manually input AT commands and the HC-05 should respond back with confirmation that the AT commands were processed correctly through the serial terminal. Using the Arduino, it required communication from the HC-05 to send to the Arduino serial terminal where it waits when serial data is available before it writes from the HC-05 to the Arduino serial terminal. Then it requires a response communication from the Arduino serial terminal to the HC-05 where it waits for serial data before it writes from the Arduino serial terminal to the HC-05. This allows continuous communication between the Arduino and the HC-05. There were also some implementations of configuring with the TM4C123 Launchpad that semi-failed but did allow some limited methods to configure. These methods will be found towards the end of the report under 'Extra' as well as a quick worded tutorial on how to configure with the Arduino serial terminal.

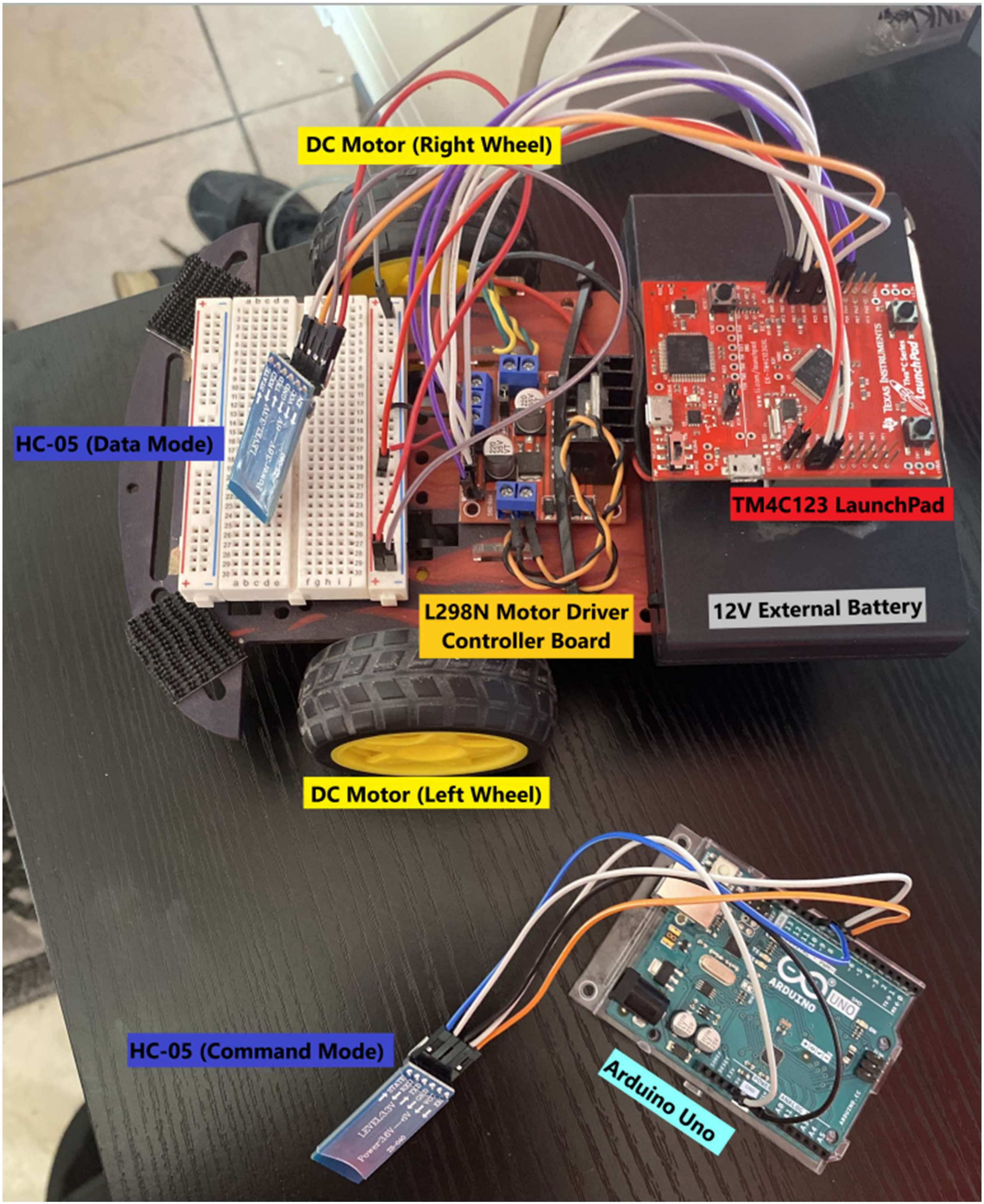
Theory for BT Controlled Robot Car:

For the BT controlled robot car, the user is supposed to send out commands through a BT terminal to control the robot car. There are 7 commands to control the car's speed, direction, and movement. There will also be LEDs that will correspond with the user's commands to determine the current state that the robot car is on. The robot car will have UART1 to allow serial communication from the HC-05 and communicate serially with the Bluetooth module. Depending on the command will determine the PWM supplied to each DC motor that control the wheels. This allows movement such as moving forward, moving backwards, doing left turns, doing right turns, stopping, speeding up, and slowing down. These commands are sent from a serial terminal named "Bluetooth Serial Terminal" and more information will be found about this terminal at the end of the report under 'Extra'.

Hardware Design/ Circuit Diagram:



Photographs of Hardware System:



Software Design for BT Configuration (Arduino Code):

```
// Name: Jesus Franco
// Student ID: 014046368
// Course Number: CECS 447
// Assignment: Project 3: Bluetooth Configuration
// Description: This code will allow the user to manually configurate the HC-05 using
//                the Arduino Serial Terminal.

//  Arduino | HC-05
//  Pin 11  | RX
//  Pin 10  | TX
//  Pin 9   | Enable
//  5v      | VCC
//  GND     | GND

#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX

void setup()
{
  pinMode(9, OUTPUT); // This pin will pull the HC-05 pin 34 (key pin) HIGH to switch
module to AT mode
  digitalWrite(9, HIGH); // Set pin 9 as logic level HIGH for HC-05 enable pin
  Serial.begin(9600); // Set serial Terminal baud rate
  Serial.println(">>> Welcome to Serial Terminal <<<"); // Serial print
  Serial.println(">>> This is the setup program for HC-05 BlueTooth Module <<<"); //
Serial print
  Serial.println(">>> You are at 'AT' Command Mode <<<"); // Serial print
  Serial.println(">>> Type 'AT' and follow with a command <<<"); // Serial print
  BTSerial.begin(38400);  // HC-05 default speed in AT command mode
}

void loop()
{

  // Keep reading from HC-05 and send to Arduino Serial Monitor
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  // Keep reading from Arduino Serial Monitor and send to HC-05
  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

Software Design for Bluetooth Controlled Robot Car:

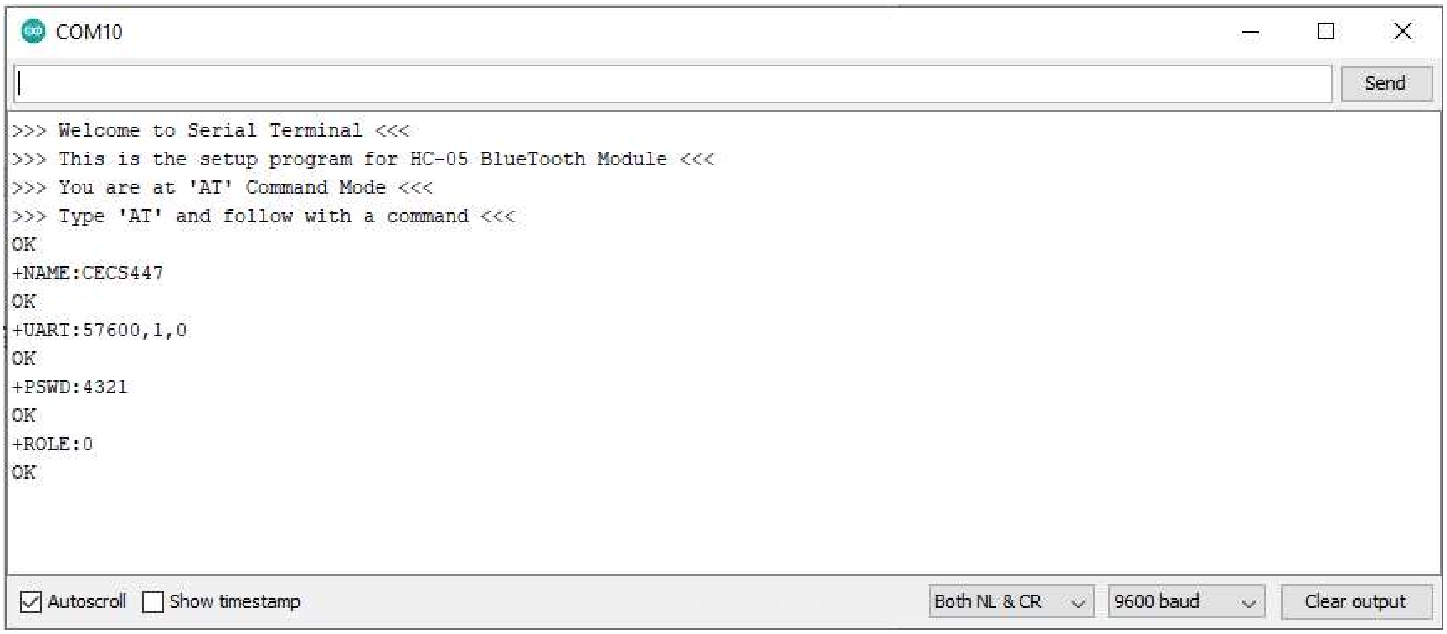
```
// Name: Jesus Franco
// Student ID: 014046368
// Course Number: CECS 447
// Assignment: Project 3: Bluetooth Controlled Car
// Description: This code will allow communication from a serial terminal to the robot
car.
//                                        There will be commands to change the speed and
direction of the car through
//                                        a Bluetooth Terminal. The UART1 configuration
is set to a baud rate of
//                                        57600 and will be located under the name of
SWAG.

//   PB3-0 -> L298N IN[4:1] respectively
//   PB6   -> L298N ENA
//   PB7   -> L298N ENB
//
//   PC4 -> UART1 RX
//   PC5 -> UART1 TX

//   Green LED    -> Forward
//   Blue LED     -> Reverse
//   Yellow LED   -> Left Turn
//   Sky Blue LED -> Right Turn
//   Blue LED             -> The car is too close to the RIGHT wall
//   No LED       -> STOP
//   Pink LED           -> Speed Up
//   Red LED                  -> Slow Down

#include "tm4c123gh6pm.h"
#include <stdint.h>
#include "PLL.h"
#include "PWM.h"
#include "UART.h"

// Constants
#define PERIOD 40000          // 3125 Hz, 100% duty cycle

// Function prototypes
void PortF_Init(void);
void PortB_Init(void);

// Global Variables
char keyPressed;                                                  // UART1 InChar
variable
unsigned long speed = 50;                             // initial speed is set to 50%

int main(void){
      PortB_Init();                                                           // Port B
initialization
      PortF_Init();                                                           // Port F
initialization
      PLL_Init();                                                             // PLL
initialization for 50MHz
      UART1_Init();

      PWM0A_Init(PERIOD, 0);          // initialize PWM0A, 3125 Hz, 0% duty
  PWM0B_Init(PERIOD, 0);              // initialize PWM0B, 3125 Hz, 0% duty
      GPIO_PORTB_DATA_R = 0x05;       // PORTB to be set to move forward initially

  while(1){
```

```
            keyPressed = UART1_InChar();  // Receive a Char variable

            switch(keyPressed){
                  case 'W':                                                         // 
Forward
                        GPIO_PORTF_DATA_R = 0x08; // GREEN
                        GPIO_PORTB_DATA_R = 0x05; // Forward direction
                        PWM0B_Duty(PERIOD * .01 * speed); // Adjust left wheel speed
                        PWM0A_Duty(PERIOD * .01 * speed); // Adjust right wheel speed
                        break;
                  case 'S':                                                         // 
Reverse
                        GPIO_PORTF_DATA_R = 0x04; // BLUE
                        GPIO_PORTB_DATA_R = 0x0A; // Reverse direction
                        PWM0B_Duty(PERIOD * .01 * speed); // Adjsut left wheel speed
                        PWM0A_Duty(PERIOD * .01 * speed); // Adjust right wheel speed
                        break;
                  case 'A':                                                         // 
Left Turn
                        GPIO_PORTF_DATA_R = 0x0A; // YELLOW
                        PWM0B_Duty(PERIOD * .20); // Adjust left wheel speed to be slower
                        PWM0A_Duty(PERIOD * .50); // Adjsut right wheel speed to be 
faster
                        break;
                  case 'D':                                                         // 
Right Turn
                        GPIO_PORTF_DATA_R = 0x0C; // SKY BLUE
                        PWM0B_Duty(PERIOD * .50); // Adjust left wheel speed to be faster
                        PWM0A_Duty(PERIOD * .20); // Adjust right wheel speed to be 
slower
                        break;
                  case 'T':                                                         // 
STOP
                        GPIO_PORTF_DATA_R = 0x00; // NONE
                        PWM0B_Duty(0);                                  // No speed
                        PWM0A_Duty(0);                                  // No speed
                        break;
                  case 'U':                                                         // 
Speed Up
                        GPIO_PORTF_DATA_R = 0x06; // PINK
                        speed = speed + 10;                       // Adjsut speed by 
increments of 10%
                        if (speed >= 100) speed = 100; // Adjust max speed to 100%
                        PWM0B_Duty(PERIOD * .01 * speed); // Adjust left wheel speed
                        PWM0A_Duty(PERIOD * .01 * speed); // Adjust right wheel speed
                        break;
                  case 'L':                                                         // 
Slow Down
                        GPIO_PORTF_DATA_R = 0x02; // RED
                        speed = speed - 10;                       // Adjust speed by 
decrements of 10%
                        if (speed <= 10) speed = 10; // Adjust min speed to be 10%
                        PWM0B_Duty(PERIOD * .01 * speed); // Adjust left wheel speed
                        PWM0A_Duty(PERIOD * .01 * speed); // Adjust right wheel speed
                        break;
            }// end switch
      } // end while loop
} // end main

// PortF_Init for initializing the onboard switches/LEDs and the interrupts
void PortF_Init(void){
      volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;     // Acativate clock for port F
```

```c
  delay = SYSCTL_RCGC2_R;           // delay
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock GPIO PortF PF0
  GPIO_PORTF_CR_R |= 0x1F;          // allow changes to PF4-0
      GPIO_PORTF_DIR_R &= ~0x11;                // PF4 & PF0 as inputs
  GPIO_PORTF_DIR_R |=  0x0E;              // PF3-0 as outputs
  GPIO_PORTF_AFSEL_R &= ~0x11;            // disable alt funct on PF4 & PF0
  GPIO_PORTF_DEN_R |= 0x1F;               // enable digital I/O on PF4 & PF0
  GPIO_PORTF_PCTL_R &= ~0x000F0000; // configure PF4 as GPIO
  GPIO_PORTF_AMSEL_R &= ~0x11;      // disable analog functionality on PF4 & PF0
  GPIO_PORTF_PUR_R |= 0x11;               // enable pull-up on PF4 & PF0
      GPIO_PORTF_IS_R &= ~0x11;                 // PF4 & PF0 is edge-sensitive
  GPIO_PORTF_IBE_R |= 0x11;                     // PF4 & PF0 is both edges
  GPIO_PORTF_IEV_R &= ~0x11;              // PF4 & PF0 falling edge event
  GPIO_PORTF_ICR_R = 0x11;                // clear flag4
  GPIO_PORTF_IM_R |= 0x11;                // arm interrupt on PF4 & PF0
  NVIC_PRI7_R |= (NVIC_PRI7_R&0xFF00FFFF)|0x00A00000; // priority 5
  NVIC_EN0_R |= 0x40000000;               // enable interrupt 30 in NVIC
}

// PortB_Init for initializing PB7-6 & PB3-0 as outputs
void PortB_Init(void){volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000002;     // Activate clock for port B
  delay = SYSCTL_RCGC2_R;           // delay
  GPIO_PORTB_DIR_R = 0x0F;          // PB3-0 as outputs
  GPIO_PORTB_DEN_R = 0x0F;          // enable digital pins PB3-PB0
}
```

Bluetooth Configuration Requirements:

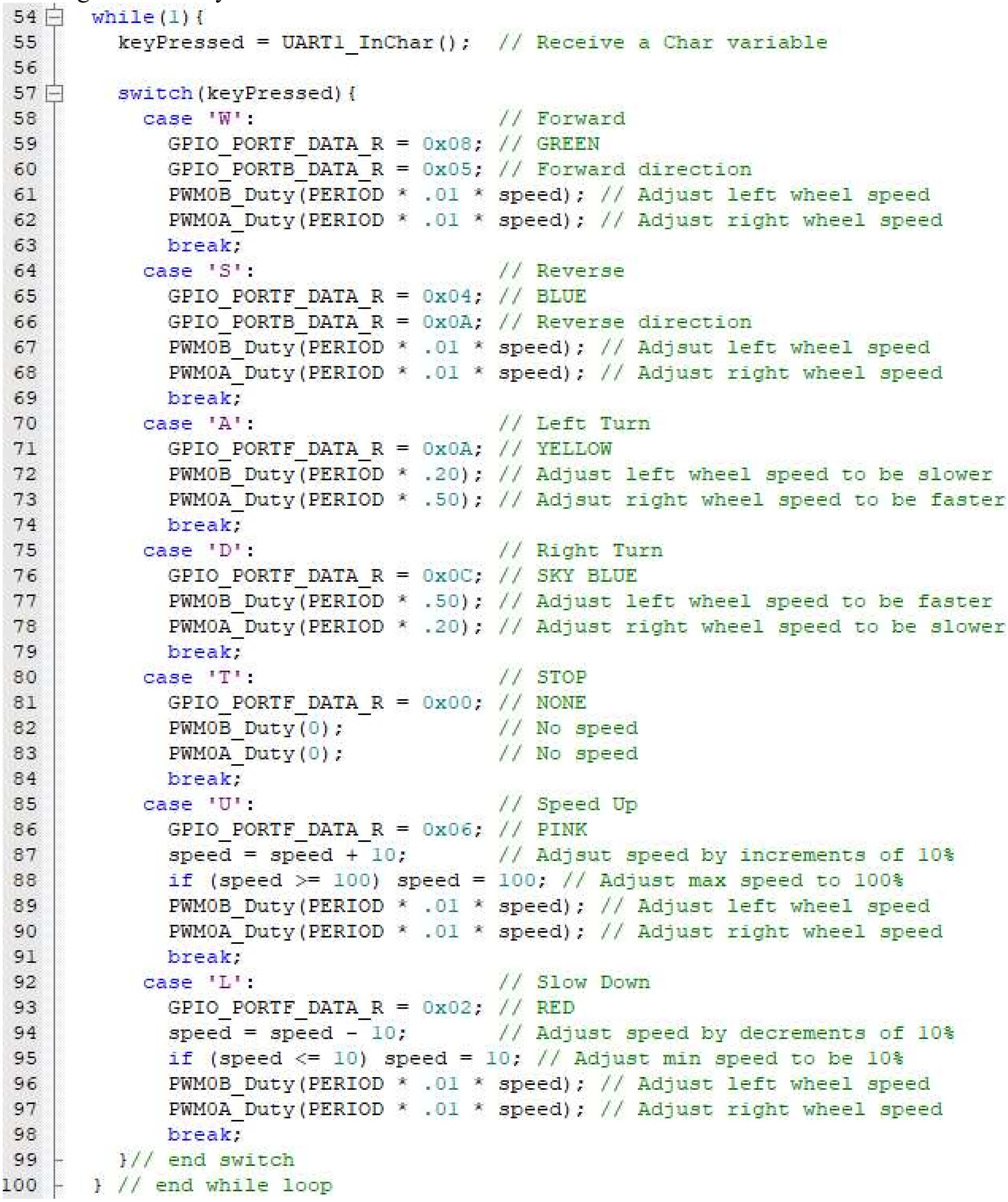
This part only required the HC-05 to be configured to this specific setup:
Name = CECS447
Baud Rate = 57600
Stop Bit = 1
Parity = 0
Password = 4321
Role = 0



```
>>> Welcome to Serial Terminal <<<
>>> This is the setup program for HC-05 BlueTooth Module <<<
>>> You are at 'AT' Command Mode <<<
>>> Type 'AT' and follow with a command <<<
OK
+NAME:CECS447
OK
+UART:57600,1,0
OK
+PSWD:4321
OK
+ROLE:0
OK
```

Bluetooth Controlled Robot Car Requirements:

This part required the car to receive a char variable by the user through a Bluetooth Terminal to control the speed, direction, and movement of the robot car using PWM. There will also be LED states to specify the current state of the robot car. The LED logic is found in the beginning of this report for more details on the meaning behind every color of the LEDs.



```c
while(1){
  keyPressed = UART1_InChar();  // Receive a Char variable

  switch(keyPressed){
    case 'W':                   // Forward
      GPIO_PORTF_DATA_R = 0x08; // GREEN
      GPIO_PORTB_DATA_R = 0x05; // Forward direction
      PWM0B_Duty(PERIOD * .01 * speed); // Adjust left wheel speed
      PWM0A_Duty(PERIOD * .01 * speed); // Adjust right wheel speed
      break;
    case 'S':                   // Reverse
      GPIO_PORTF_DATA_R = 0x04; // BLUE
      GPIO_PORTB_DATA_R = 0x0A; // Reverse direction
      PWM0B_Duty(PERIOD * .01 * speed); // Adjsut left wheel speed
      PWM0A_Duty(PERIOD * .01 * speed); // Adjust right wheel speed
      break;
    case 'A':                   // Left Turn
      GPIO_PORTF_DATA_R = 0x0A; // YELLOW
      PWM0B_Duty(PERIOD * .20); // Adjust left wheel speed to be slower
      PWM0A_Duty(PERIOD * .50); // Adjsut right wheel speed to be faster
      break;
    case 'D':                   // Right Turn
      GPIO_PORTF_DATA_R = 0x0C; // SKY BLUE
      PWM0B_Duty(PERIOD * .50); // Adjust left wheel speed to be faster
      PWM0A_Duty(PERIOD * .20); // Adjust right wheel speed to be slower
      break;
    case 'T':                   // STOP
      GPIO_PORTF_DATA_R = 0x00; // NONE
      PWM0B_Duty(0);            // No speed
      PWM0A_Duty(0);            // No speed
      break;
    case 'U':                   // Speed Up
      GPIO_PORTF_DATA_R = 0x06; // PINK
      speed = speed + 10;       // Adjsut speed by increments of 10%
      if (speed >= 100) speed = 100; // Adjust max speed to 100%
      PWM0B_Duty(PERIOD * .01 * speed); // Adjust left wheel speed
      PWM0A_Duty(PERIOD * .01 * speed); // Adjust right wheel speed
      break;
    case 'L':                   // Slow Down
      GPIO_PORTF_DATA_R = 0x02; // RED
      speed = speed - 10;       // Adjust speed by decrements of 10%
      if (speed <= 10) speed = 10; // Adjust min speed to be 10%
      PWM0B_Duty(PERIOD * .01 * speed); // Adjust left wheel speed
      PWM0A_Duty(PERIOD * .01 * speed); // Adjust right wheel speed
      break;
  }// end switch
} // end while loop
```

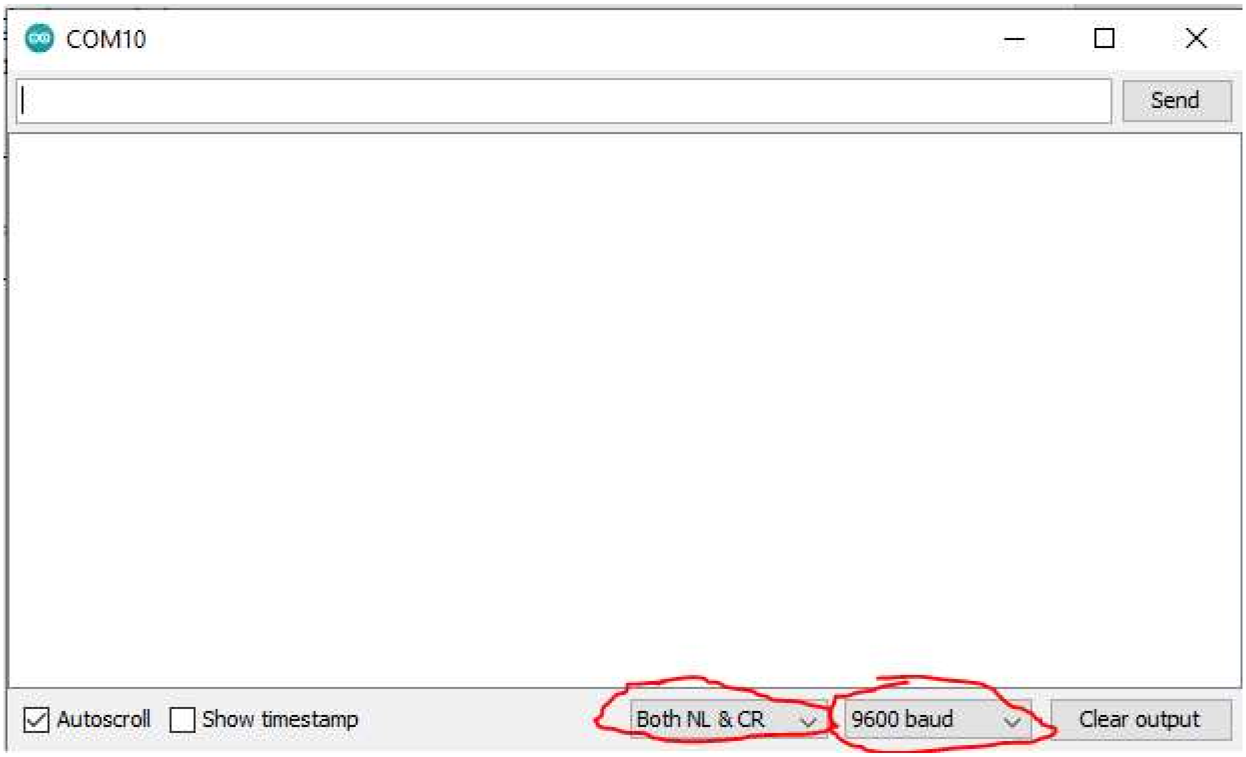
Conclusion:

For this project, the most difficult part was to configure the HC-05. After some semi-failed attempts that will be discussed at the end of this report under ‘Extra’, I have managed to successfully configure this HC-05 using the Arduino. This was very handy since the Arduino also came with its own serial terminal that was very simple to allow AT commands to be outputting to the HC-05; thus, configuring it successfully. After the HC-05 was configured, the Bluetooth controlled robot car was very clear to implement since the robot car was built from the previous class and most of the code is very similar to the final project of the previous class. This just required a switch case statement to determine the state of the robot car to control to control the PWM.

Extra:

Worded Tutorial to Configure the HC-05 with the Arduino:

1. After the Arduino code is compiled and uploaded onto the Arduino board, these steps are necessary:
2. Power off the Arduino.
3. Disconnect the HC-05 VCC pin from the Arduino (since the Arduino does not have a ON/OFF switch).
4. Connect the Arduino.
5. Hold reset on the HC-05 while connecting the VCC pin back to the Arduino to put it in command mode.
6. Next, click the magnifying glass icon to access the Arduino serial terminal.
7. Make sure that ‘Both NL & CR’ is selected to allow a new line and carry return. Also make sure that the serial terminal baud rate is set to ‘9600 baud’ since the Arduino code is set to 9600.

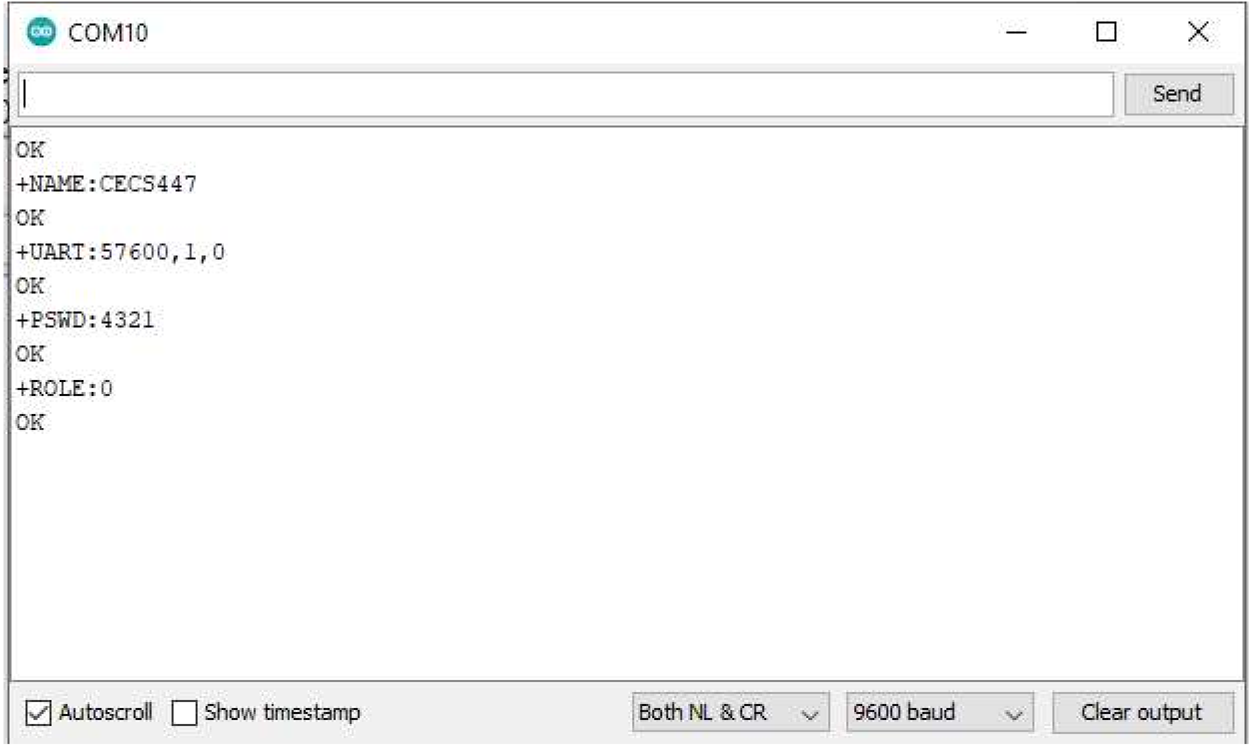


☑ Autoscroll □ Show timestamp

Both NL & CR

9600 baud

8. After that step is complete, you can start typing AT commands. The following images show the confirmation of the desired setup by typing AT commands ‘AT’, ‘AT+NAME?’, ‘AT+UART?’, ‘AT+PSWD?’, and ‘AT+ROLE’.



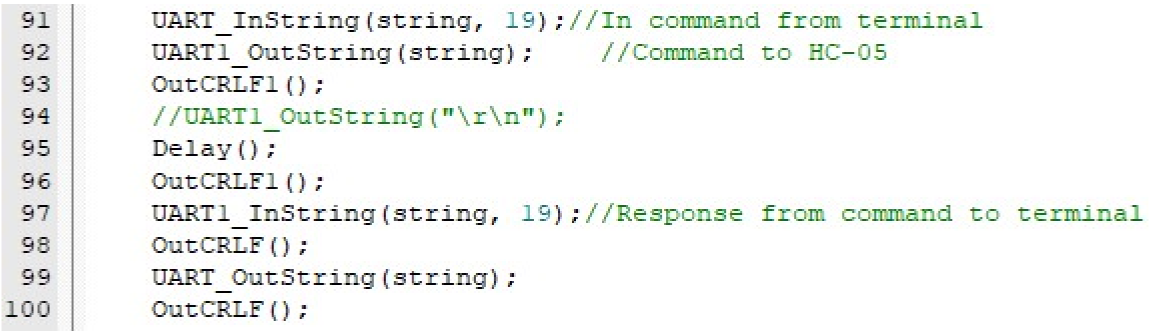
```
OK
+NAME:CECS447
OK
+UART:57600,1,0
OK
+PSWD:4321
OK
+ROLE:0
OK
```

☑ Autoscroll □ Show timestamp Both NL & CR 9600 baud

Method 1 to Configure the HC-05 with the TM4C123 Launchpad:

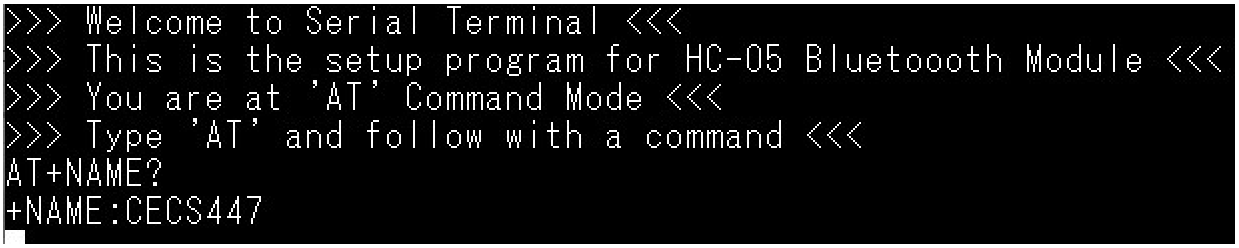
This method semi-failed because it only allowed one AT command to successfully go through. This is still able to configure the HC-05 but will required to constantly keep resetting since it only works once.

Code:



```
    UART_InString(string, 19);//In command from terminal
    UART1_OutString(string);    //Command to HC-05
    OutCRLF1();
    //UART1_OutString("\r\n");
    Delay();
    OutCRLF1();
    UART1_InString(string, 19);//Response from command to terminal
    OutCRLF();
    UART_OutString(string);
    OutCRLF();
```

TeraTerm:



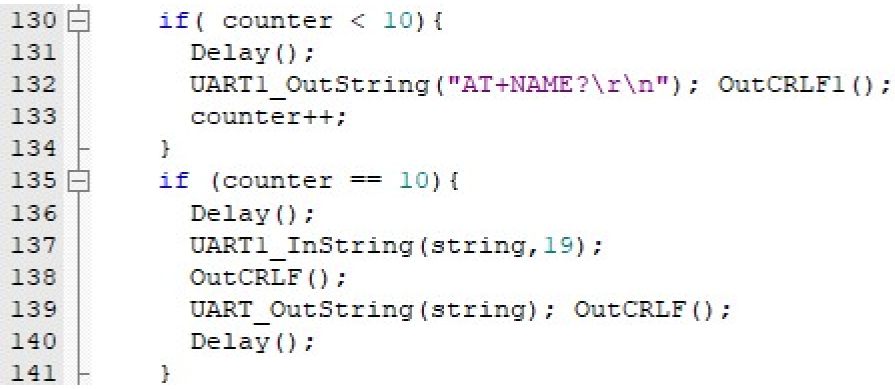
```
>>> Welcome to Serial Terminal <<<
>>> This is the setup program for HC-05 Bluetoooth Module <<<
>>> You are at 'AT' Command Mode <<<
>>> Type 'AT' and follow with a command <<<
AT+NAME?
+NAME:CECS447
```

This method will work once and then crash and will not allow any user input. This method successfully works and allows at least 1 AT command to go through, so it is possible to use, but not efficient compared to the Arduino method.

Method 2 to Configure the HC-05 with the TM4C123 Launchpad:

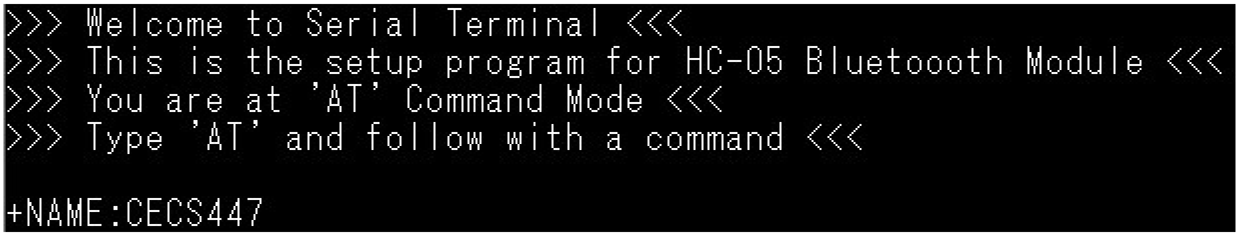
This method also semi-failed because it also only allows at least one AT command to go through, but this time instead of manually inputting the command, this method will just output the string and check it.

Code:



```
    if( counter < 10){
      Delay();
      UART1_OutString("AT+NAME?\r\n"); OutCRLF1();
      counter++;
    }
    if (counter == 10){
      Delay();
      UART1_InString(string,19);
      OutCRLF();
      UART_OutString(string); OutCRLF();
      Delay();
    }
```

TeraTerm:

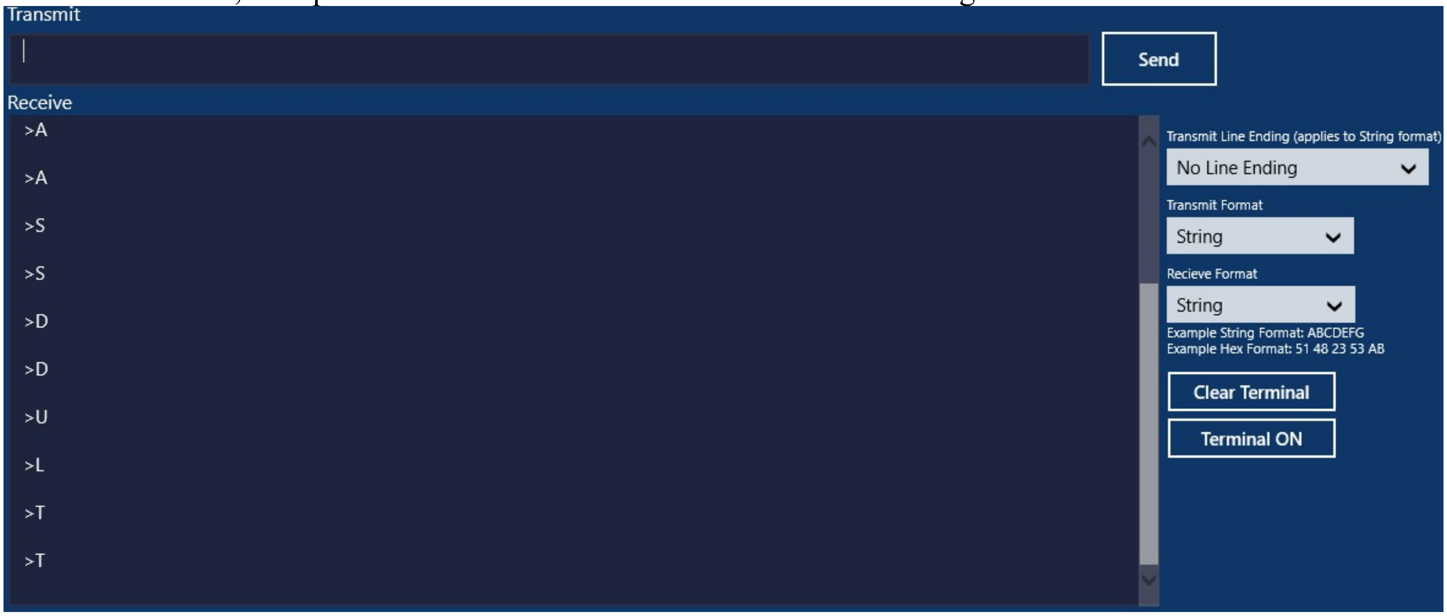


```
>>> Welcome to Serial Terminal <<<
>>> This is the setup program for HC-05 Bluetoooth Module <<<
>>> You are at 'AT' Command Mode <<<
>>> Type 'AT' and follow with a command <<<

+NAME:CECS447
```

This method also only works once and will not allow a second command to go through either. Still successfully works but completely outclassed and inefficient compared to the Arduino method.

Bluetooth Serial Terminal:

The Bluetooth serial terminal that I used is a Microsoft app that is named “Bluetooth Serial Terminal”. To use this Bluetooth terminal, once must pair the Bluetooth device directly from the computer’s settings. Once it is paired, one can open the Bluetooth serial terminal app and scroll down the list to find the device that would use the serial terminal, then press ‘connect’. The user can now start transmitting data to the HC-05.



```
>A
>A
>S
>S
>D
>D
>U
>L
>T
>T
```

Transmit Line Ending (applies to String format): No Line Ending

Transmit Format: String

Recieve Format: String

Example String Format: ABCDEFG
Example Hex Format: 51 48 23 53 AB

Here it shows the char values that have been transmitted and it shows it being received. This terminal is very easy to use and can send string and hex values as well. The link to this app will be provided below:

https://www.microsoft.com/en-us/p/bluetooth-serial-terminal/9wzdncrdfst8?activetab=pivot:overviewtab