CECS 447 Fall 2022 Project # 3

Bluetooth Controlled Robot Car

By

Abhishek Jasti, Anand Jasti

October 31, 2022

Design and build a robot car with GPIO, hardware PWM, UART, power supply, HC-05 Bluetooth module, DC motors, and L298N motor drive controller using TCM4C123G Launchpad Microcontroller.

CECS 447 Project 3 Report

Introduction

In this project we created a connection between laptop/smartphone and the robot car using Bluetooth. This robot car will be able to go forwards, backwards, speed up, slow down, and make turns. These actions are controlled by laptop/smartphone via Bluetooth. The speed up and slowing down of the robot car is done by implementing hardware PWM. There are two source codes for this project, one for setting up the Bluetooth module and the other for controlling the Bluetooth module. The Bluetooth module will be setup through the serial terminal, and the control of the Bluetooth module will be done through a Bluetooth terminal app on the laptop/smartphone. We built this project using basic hardware components is our embedded systems, while using GPIO, hardware PWM, UART, power supply, HC-05 Bluetooth module, DC motors, and L298N motor drive controller. Our goal for this project is to use UART protocol communication between a laptop/smartphone and the microcontroller.

Operation

When the setup source code is run, the system will show the user a welcome message and an example text on how to setup the HC-05 Bluetooth module on the serial terminal of the laptop. The Bluetooth module should be in command mode before we can set up our module. To get the HC-05 Bluetooth module in command mode, we want to connect the EN pin to 3.3v and hold down the on-board push button before turning on the module. To setup the HC-05 Bluetooth module the user can change the name, the baud rate, the four-digit passcode, and the role of the HC-05 Bluetooth module.

When the control source code is run, the EN pin on the Bluetooth module should be disconnected. To connect the Bluetooth module to the laptop/smartphone we first turn on the Bluetooth module and go to the settings of our smart device. In settings we turn on our smart device’s Bluetooth and look for the name of our HC-05 Bluetooth module (which should be the name that was setup during the setup process). Then on the smart device, after finding the name of our Bluetooth module, we click it and put in the four-digit passcode (which should be the passcode that was setup during the setup process). After entering the passcode, the Bluetooth module will be connected to the smart device, then we open the Bluetooth terminal app on the smart device (this app should be installed beforehand). After opening the Bluetooth terminal app, we first choose the name of our Bluetooth module again and then press the connect button in the Bluetooth terminal app. Now we can start controlling the robot car, there are seven commands (which are represented as single characters) that we can use to control the robot car. These commands and their actions are listed below.

Command	Robot car action	On-board LED light
‘F’	Forward	Green
‘B’	Reverse	Blue
‘L’	Left Turn	Yellow
‘R’	Right Turn	Purple
‘S’	Stop	No LED
‘U’	Speed Up	Direction LED
‘D’	Slow Down	Direction LED

Link to Setup Demonstration video:
https://drive.google.com/file/d/1wlEUN06VO3LiJAnqZnCrDsEkPn-tXkXV/view?usp=sharing

Link to Control Demonstration video:
https://drive.google.com/file/d/1W2gVzAbADAYUrXsUdk9pMzzSYpPe9Xmb/view?usp=sharing

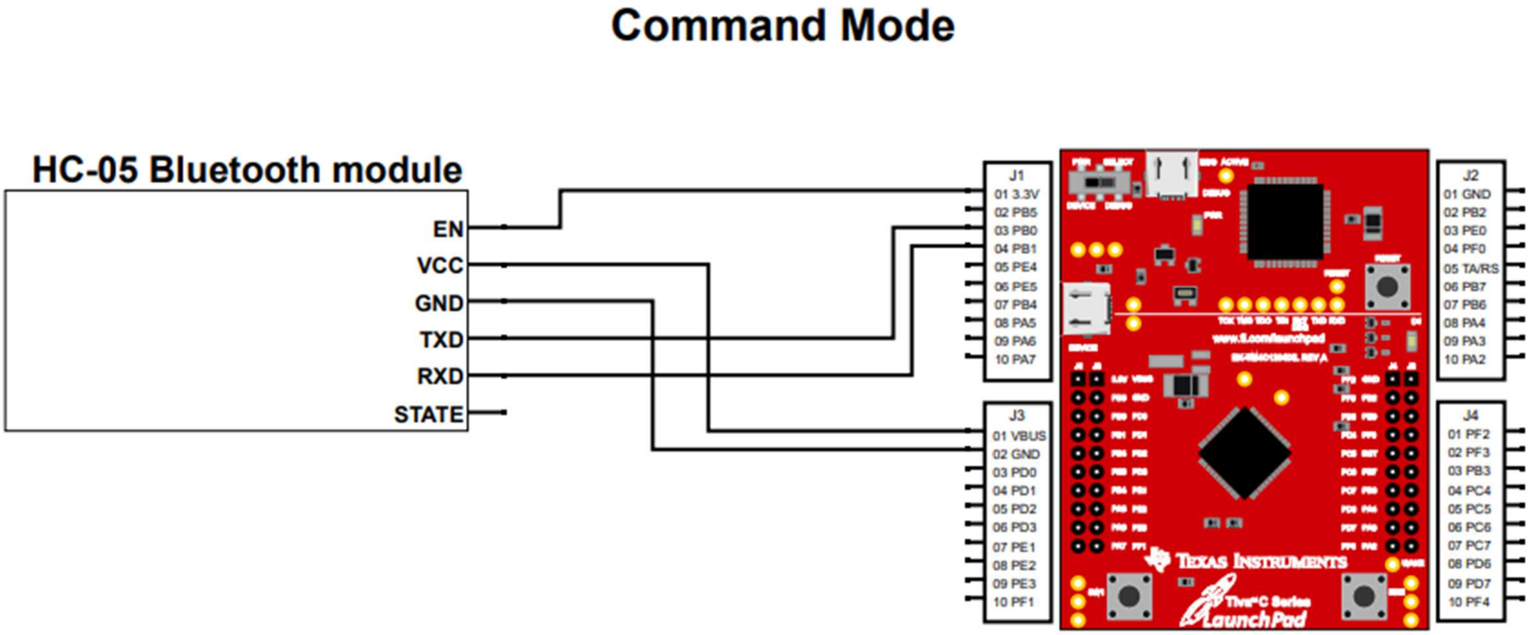
Theory

This project uses ARM Cortex TM4C123GH6PM Microcontroller, more specifically we used three of the six Genera-Purpose I/O ports (PA, PB, and PF). In port A we used six pins to connect to the L298N motor drive controller (PA2 through PA7). In port A we also used two internal pins to connect to the serial terminal of the computer using UART 0 with pins PA0, and PA1. In port B we used two pins to connect to the HC-05 Bluetooth module using UART 1 with pins PB0, and PB1. In port F we used three pins that connect to the on-board Red, Blue, and Green LEDs (more specifically defined in the hardware design). In this project we created driver functions that were needed to communicate between the Bluetooth module and the microcontroller. A lot of the driver functions used in this project were provided to us before hand, we only had to change a little bit of code to accurately perform the requirements for this project (more specifically defined in software design).

Hardware design

Communication/Setup Mode:

Schematic:



Command Mode

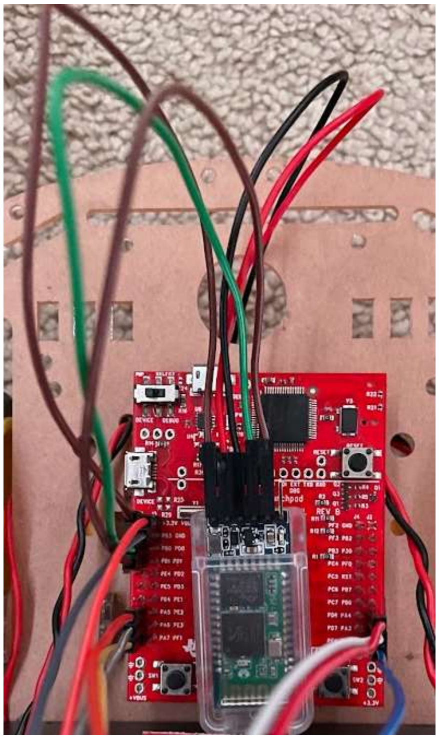
Outputs:

Connects to computer (Internal connection via USB to U0Tx)	PA1
Connects to HC-05 Bluetooth module pin RXD (U1Tx -> RXD)	PB1

Inputs:

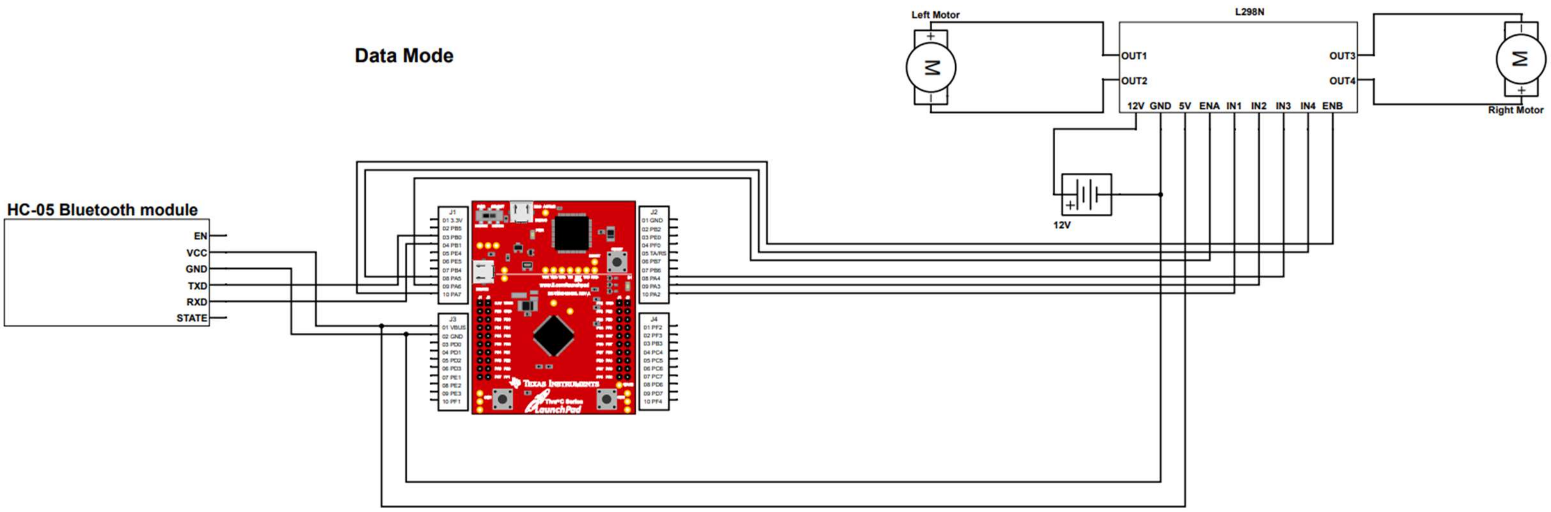
Connects to computer (Internal connection via USB to U0Rx)	**PA0**
Connects to HC-05 Bluetooth module pin TXD (U1Rx -> TXD)	**PB0**

Pictures of Hardware System:



Data/Control Mode:

Schematic:



Outputs:

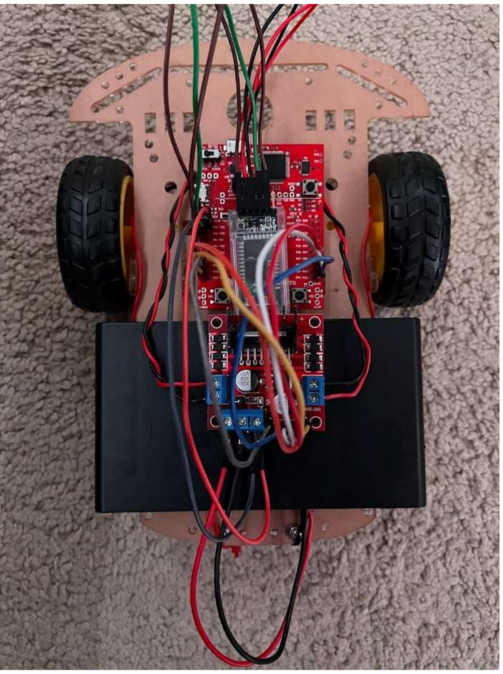
Connects to computer (Internal connection via USB to U0Tx)	PA1
Connects to HC-05 Bluetooth module pin RXD (U1Tx -> RXD)	PB1
Connects to Red LED	PF1
Connects to Blue LED	PF2
Connects to Green LED	PF3
Connects to pin 7 (ENA on L298N)	PA6
Connects of pin 8 (IN1 on L298N)	PA2
Connects of pin 9 (IN2 on L298N)	PA3
Connects of pin 10 (IN3 on L298N)	PA4
Connects of pin 11 (IN4 on L298N)	PA5
Connects of pin 12 (ENB on L298N)	PA7

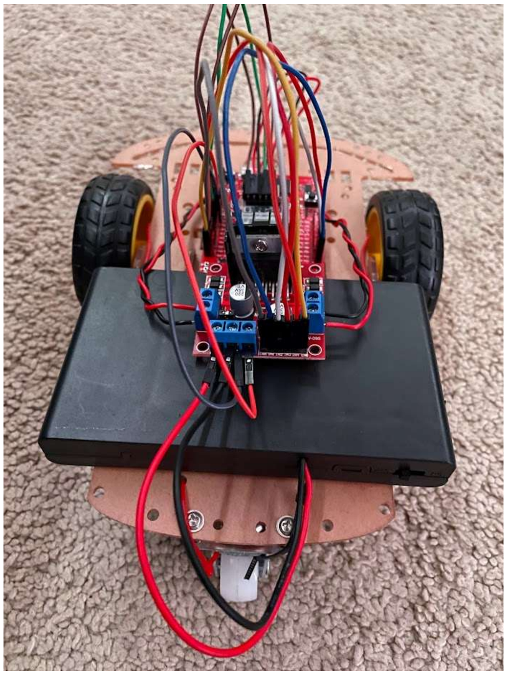
Inputs:

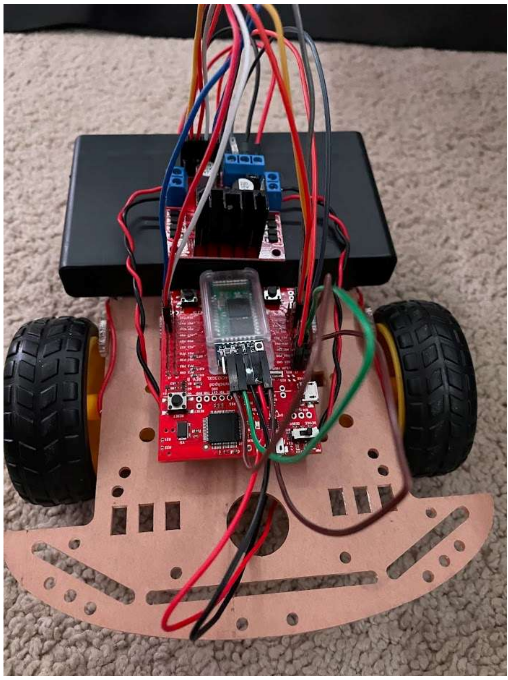
Connects to computer (Internal connection via USB to U0Rx)	**PA0**
Connects to HC-05 Bluetooth module pin TXD (U1Rx -> TXD)	**PB0**

Red, Blue, and Green LEDs are on-board LEDs provided by the launchpad.

Pictures of Hardware System:



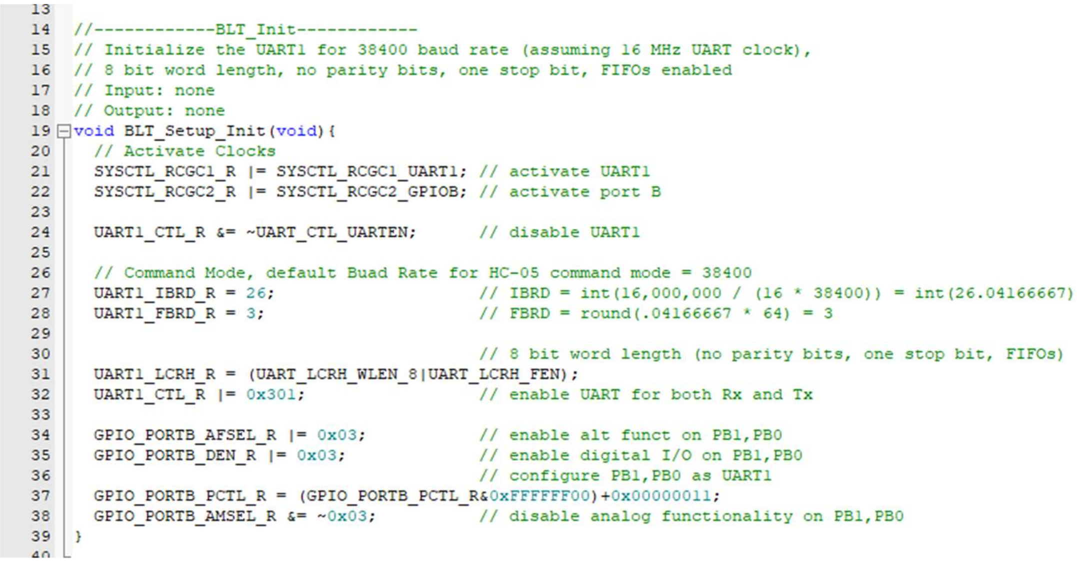




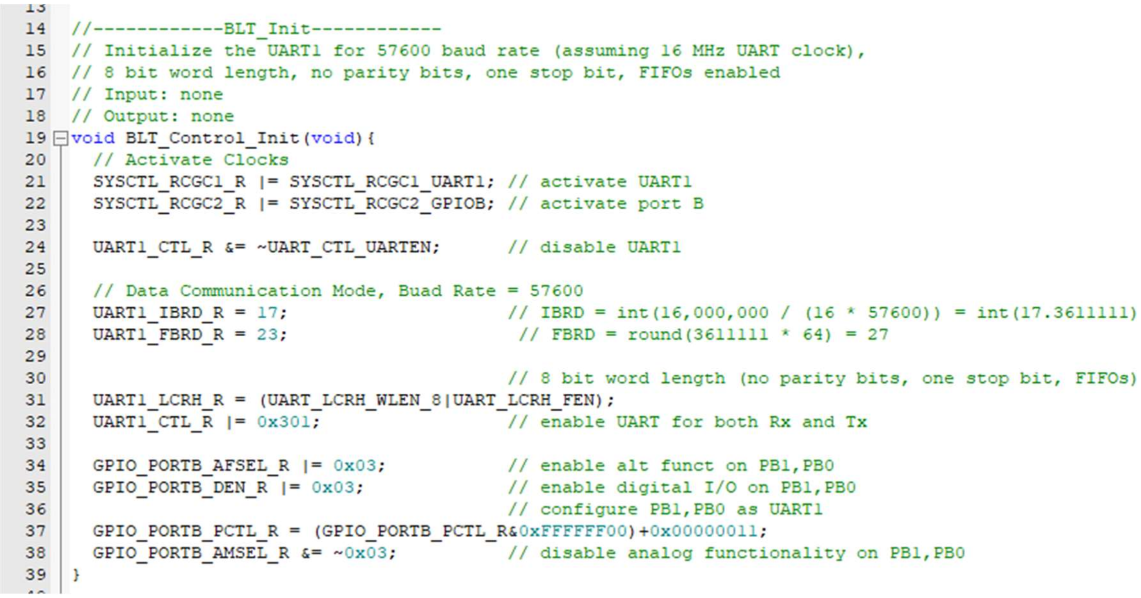
Software design

Software Source Code:

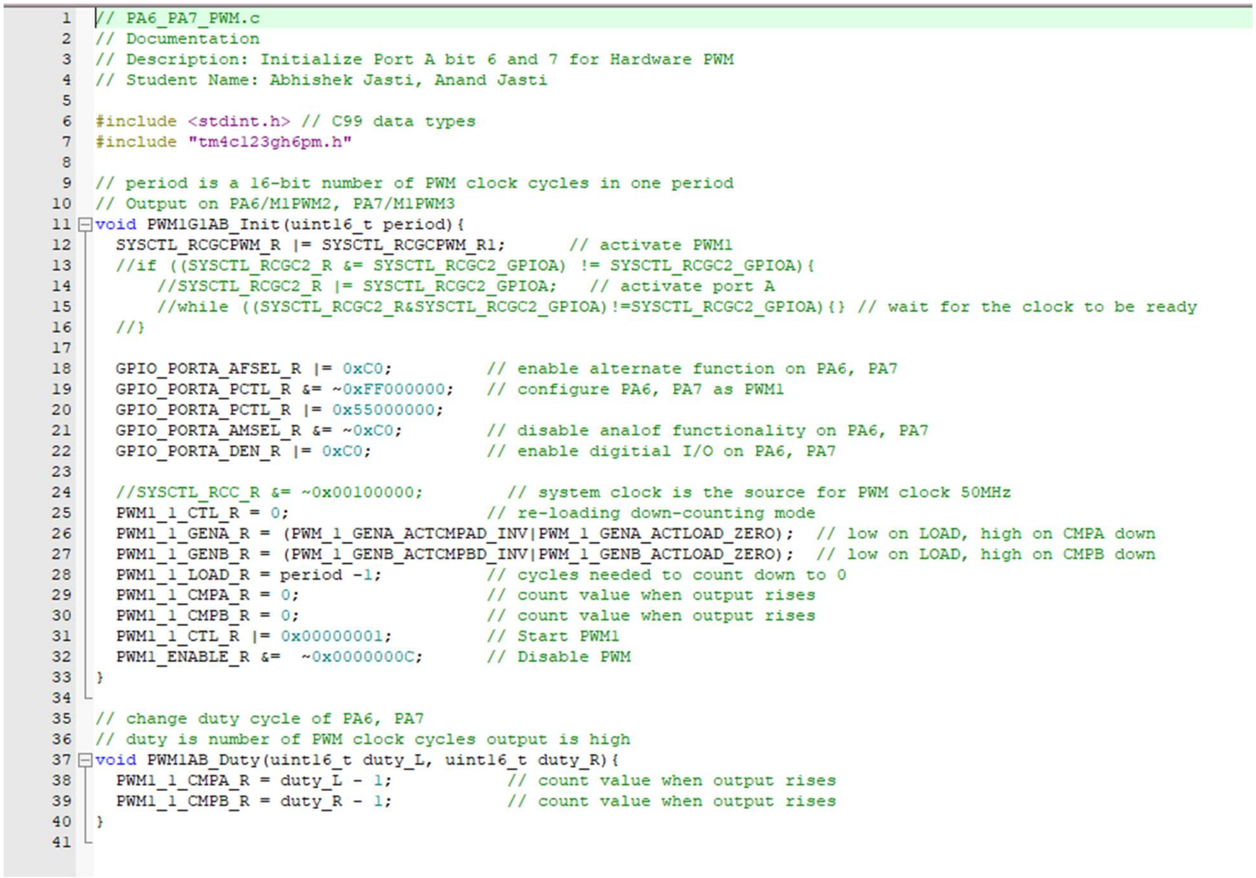
UART.c, BLT_Setup.c, and BLT_Control.c files were provided for us. UART.c was used to provide communication between the serial terminal of the laptop and the microcontroller. Both BLT_Setup.c and BLT_Control.c files are used to provided communication between the HC-05 Bluetooth module and the microcontroller. We created the BLT_control.c file based upon the BLT_Setup.c file the only difference between these two files is the initialization of the uart baud rate.



```c
//------------BLT_Init------------
// Initialize the UART1 for 38400 baud rate (assuming 16 MHz UART clock),
// 8 bit word length, no parity bits, one stop bit, FIFOs enabled
// Input: none
// Output: none
void BLT_Setup_Init(void){
  // Activate Clocks
  SYSCTL_RCGC1_R |= SYSCTL_RCGC1_UART1; // activate UART1
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOB; // activate port B

  UART1_CTL_R &= ~UART_CTL_UARTEN;      // disable UART1

  // Command Mode, default Buad Rate for HC-05 command mode = 38400
  UART1_IBRD_R = 26;                    // IBRD = int(16,000,000 / (16 * 38400)) = int(26.04166667)
  UART1_FBRD_R = 3;                     // FBRD = round(.04166667 * 64) = 3

                                        // 8 bit word length (no parity bits, one stop bit, FIFOs)
  UART1_LCRH_R = (UART_LCRH_WLEN_8|UART_LCRH_FEN);
  UART1_CTL_R |= 0x301;                 // enable UART for both Rx and Tx

  GPIO_PORTB_AFSEL_R |= 0x03;           // enable alt funct on PB1,PB0
  GPIO_PORTB_DEN_R |= 0x03;             // enable digital I/O on PB1,PB0
                                        // configure PB1,PB0 as UART1
  GPIO_PORTB_PCTL_R = (GPIO_PORTB_PCTL_R&0xFFFFFF00)+0x00000011;
  GPIO_PORTB_AMSEL_R &= ~0x03;          // disable analog functionality on PB1,PB0
}
```

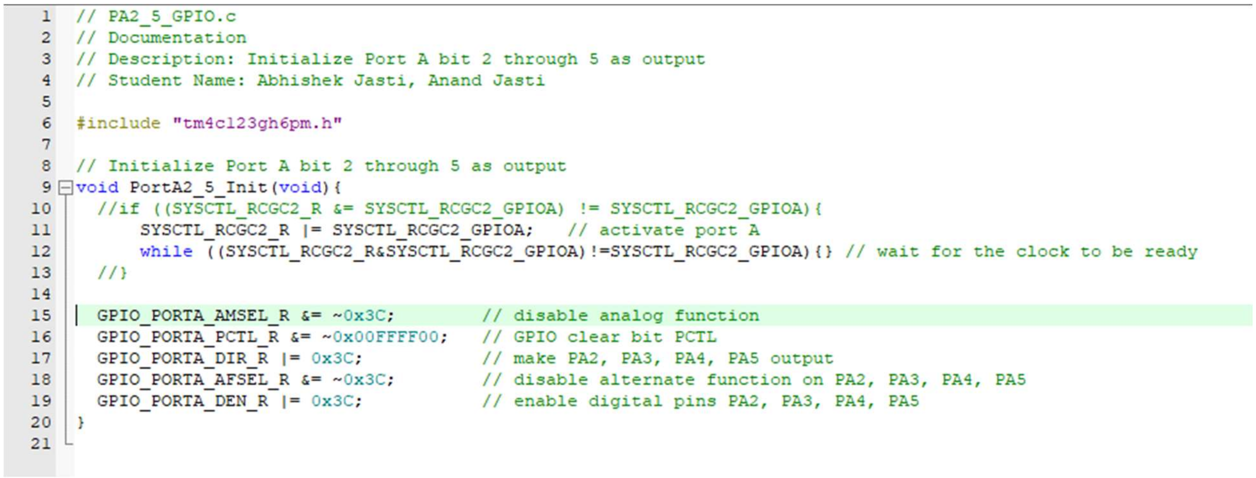


```c
//------------BLT_Init------------
// Initialize the UART1 for 57600 baud rate (assuming 16 MHz UART clock),
// 8 bit word length, no parity bits, one stop bit, FIFOs enabled
// Input: none
// Output: none
void BLT_Control_Init(void){
  // Activate Clocks
  SYSCTL_RCGC1_R |= SYSCTL_RCGC1_UART1; // activate UART1
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOB; // activate port B

  UART1_CTL_R &= ~UART_CTL_UARTEN;      // disable UART1

  // Data Communication Mode, Buad Rate = 57600
  UART1_IBRD_R = 17;                    // IBRD = int(16,000,000 / (16 * 57600)) = int(17.3611111)
  UART1_FBRD_R = 23;                     // FBRD = round(3611111 * 64) = 27

                                        // 8 bit word length (no parity bits, one stop bit, FIFOs)
  UART1_LCRH_R = (UART_LCRH_WLEN_8|UART_LCRH_FEN);
  UART1_CTL_R |= 0x301;                 // enable UART for both Rx and Tx

  GPIO_PORTB_AFSEL_R |= 0x03;           // enable alt funct on PB1,PB0
  GPIO_PORTB_DEN_R |= 0x03;             // enable digital I/O on PB1,PB0
                                        // configure PB1,PB0 as UART1
  GPIO_PORTB_PCTL_R = (GPIO_PORTB_PCTL_R&0xFFFFFF00)+0x00000011;
  GPIO_PORTB_AMSEL_R &= ~0x03;          // disable analog functionality on PB1,PB0
}
```



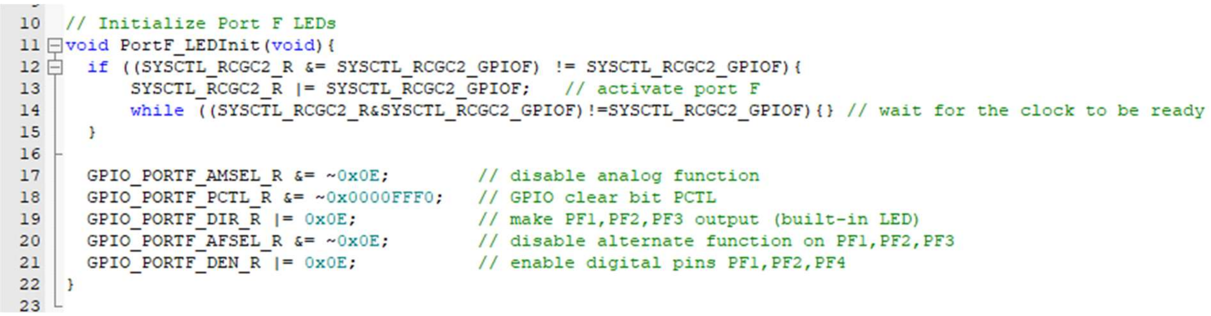
```c
// PA6_PA7_PWM.c
// Documentation
// Description: Initialize Port A bit 6 and 7 for Hardware PWM
// Student Name: Abhishek Jasti, Anand Jasti

#include <stdint.h> // C99 data types
#include "tm4c123gh6pm.h"

// period is a 16-bit number of PWM clock cycles in one period
// Output on PA6/M1PWM2, PA7/M1PWM3
void PWM1G1AB_Init(uint16_t period){
  SYSCTL_RCGCPWM_R |= SYSCTL_RCGCPWM_R1;      // activate PWM1
  //if ((SYSCTL_RCGC2_R &= SYSCTL_RCGC2_GPIOA) != SYSCTL_RCGC2_GPIOA){
      //SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOA;   // activate port A
      //while ((SYSCTL_RCGC2_R&SYSCTL_RCGC2_GPIOA)!=SYSCTL_RCGC2_GPIOA){} // wait for the clock to be ready
  //}

  GPIO_PORTA_AFSEL_R |= 0xC0;          // enable alternate function on PA6, PA7
  GPIO_PORTA_PCTL_R &= ~0xFF000000;    // configure PA6, PA7 as PWM1
  GPIO_PORTA_PCTL_R |= 0x55000000;
  GPIO_PORTA_AMSEL_R &= ~0xC0;         // disable analof functionality on PA6, PA7
  GPIO_PORTA_DEN_R |= 0xC0;            // enable digitial I/O on PA6, PA7

  //SYSCTL_RCC_R &= ~0x00100000;         // system clock is the source for PWM clock 50MHz
  PWM1_1_CTL_R = 0;                    // re-loading down-counting mode
  PWM1_1_GENA_R = (PWM_1_GENA_ACTCMPAD_INV|PWM_1_GENA_ACTLOAD_ZERO);  // low on LOAD, high on CMPA down
  PWM1_1_GENB_R = (PWM_1_GENB_ACTCMPBD_INV|PWM_1_GENB_ACTLOAD_ZERO);  // low on LOAD, high on CMPB down
  PWM1_1_LOAD_R = period -1;           // cycles needed to count down to 0
  PWM1_1_CMPA_R = 0;                   // count value when output rises
  PWM1_1_CMPB_R = 0;                   // count value when output rises
  PWM1_1_CTL_R |= 0x00000001;          // Start PWM1
  PWM1_ENABLE_R &=  ~0x0000000C;       // Disable PWM
}

// change duty cycle of PA6, PA7
// duty is number of PWM clock cycles output is high
void PWM1AB_Duty(uint16_t duty_L, uint16_t duty_R){
  PWM1_1_CMPA_R = duty_L - 1;           // count value when output rises
  PWM1_1_CMPB_R = duty_R - 1;           // count value when output rises
}
```

In this bit of code above we are initializing Port A bit 6 and 7 for Hardware PWM, and we are also providing a function to change the duty cycle of PA6 and PA7. We are using these two pins to control the speed of the left and right DC motors using the L298N motor drive controller.



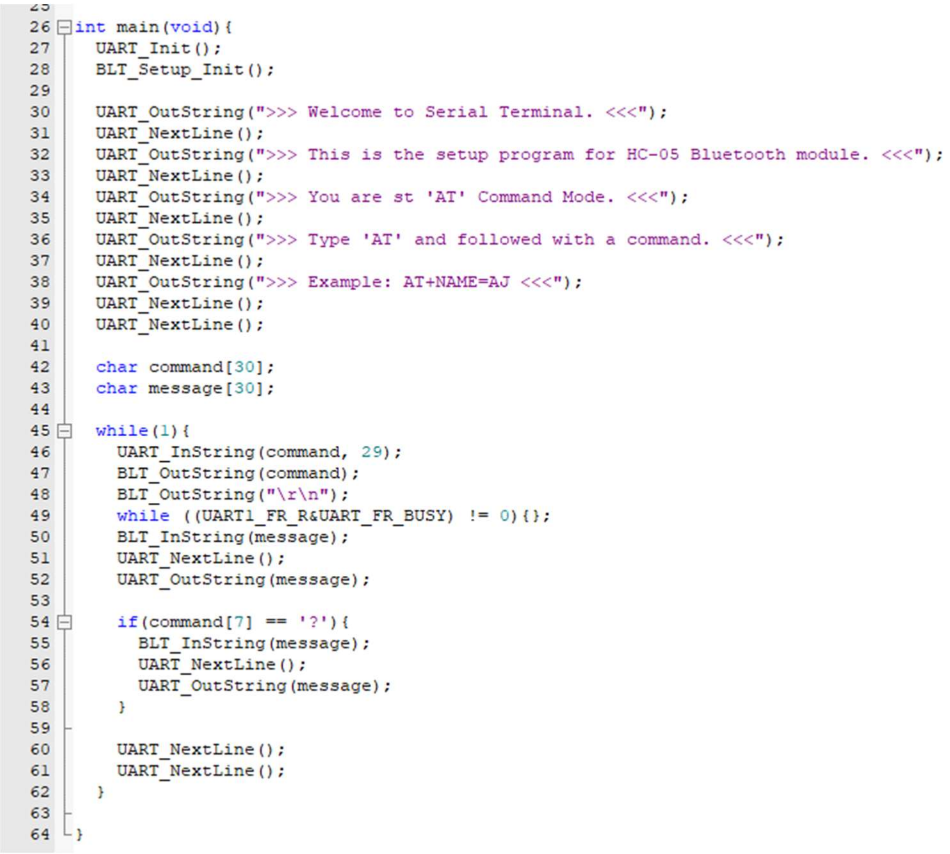
```c
// PA2_5_GPIO.c
// Documentation
// Description: Initialize Port A bit 2 through 5 as output
// Student Name: Abhishek Jasti, Anand Jasti

#include "tm4c123gh6pm.h"

// Initialize Port A bit 2 through 5 as output
void PortA2_5_Init(void){
  //if ((SYSCTL_RCGC2_R &= SYSCTL_RCGC2_GPIOA) != SYSCTL_RCGC2_GPIOA){
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOA;   // activate port A
      while ((SYSCTL_RCGC2_R&SYSCTL_RCGC2_GPIOA)!=SYSCTL_RCGC2_GPIOA){} // wait for the clock to be ready
  //}

  GPIO_PORTA_AMSEL_R &= ~0x3C;         // disable analog function
  GPIO_PORTA_PCTL_R &= ~0x00FFFF00;    // GPIO clear bit PCTL
  GPIO_PORTA_DIR_R |= 0x3C;            // make PA2, PA3, PA4, PA5 output
  GPIO_PORTA_AFSEL_R &= ~0x3C;         // disable alternate function on PA2, PA3, PA4, PA5
  GPIO_PORTA_DEN_R |= 0x3C;            // enable digital pins PA2, PA3, PA4, PA5
}
```

In this code above we are initializing Port A pins two through five to be outputs. We are using these four pins to control the direction of the left and right DC motors using the L298N motor drive controller.



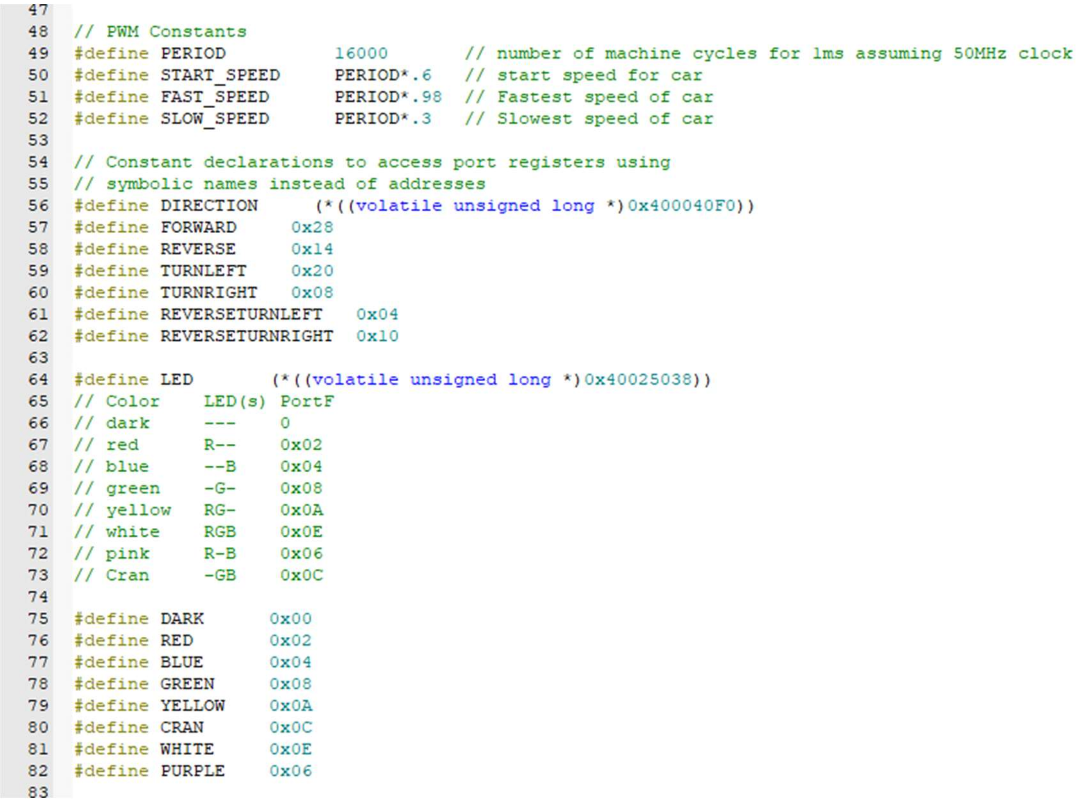
```
// Initialize Port F LEDs
void PortF_LEDInit(void){
  if ((SYSCTL_RCGC2_R &= SYSCTL_RCGC2_GPIOF) != SYSCTL_RCGC2_GPIOF){
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOF;   // activate port F
      while ((SYSCTL_RCGC2_R&SYSCTL_RCGC2_GPIOF)!=SYSCTL_RCGC2_GPIOF){} // wait for the clock to be ready
  }

  GPIO_PORTF_AMSEL_R &= ~0x0E;        // disable analog function
  GPIO_PORTF_PCTL_R &= ~0x0000FFF0;   // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R |= 0x0E;           // make PF1,PF2,PF3 output (built-in LED)
  GPIO_PORTF_AFSEL_R &= ~0x0E;        // disable alternate function on PF1,PF2,PF3
  GPIO_PORTF_DEN_R |= 0x0E;           // enable digital pins PF1,PF2,PF4
}
```

In the bit of code above we are initializing Port F pins one, two, and three. We are using these three pins as outputs (which are built-in LEDs). In this project we are using the built-in LEDs to indicate the status of the robot car. The LED status color information is listed in Operation.



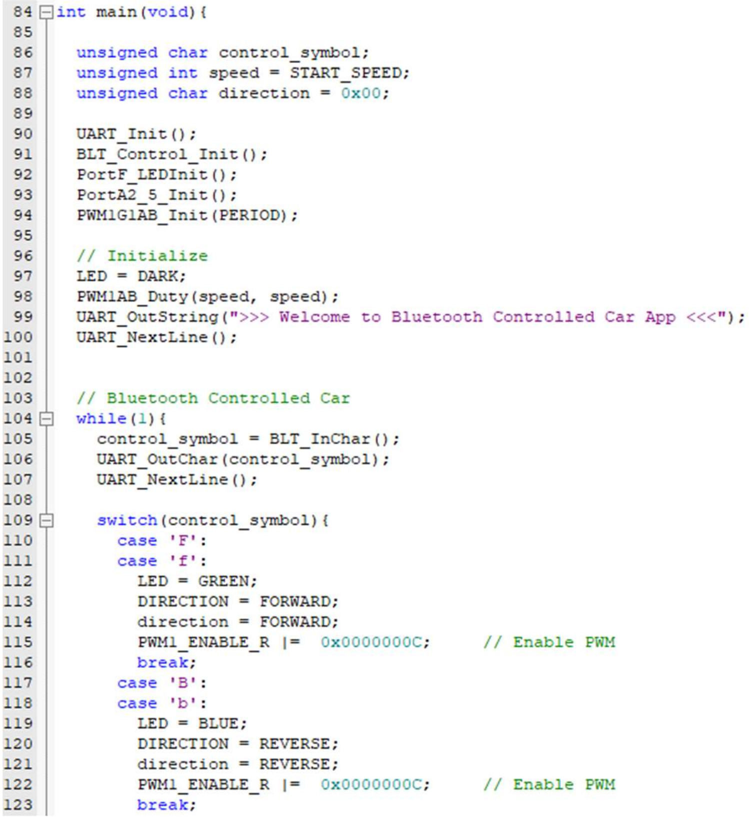
```
int main(void){
  UART_Init();
  BLT_Setup_Init();

  UART_OutString(">>> Welcome to Serial Terminal. <<<");
  UART_NextLine();
  UART_OutString(">>> This is the setup program for HC-05 Bluetooth module. <<<");
  UART_NextLine();
  UART_OutString(">>> You are st 'AT' Command Mode. <<<");
  UART_NextLine();
  UART_OutString(">>> Type 'AT' and followed with a command. <<<");
  UART_NextLine();
  UART_OutString(">>> Example: AT+NAME=AJ <<<");
  UART_NextLine();
  UART_NextLine();

  char command[30];
  char message[30];

  while(1){
    UART_InString(command, 29);
    BLT_OutString(command);
    BLT_OutString("\r\n");
    while ((UART1_FR_R&UART_FR_BUSY) != 0){};
    BLT_InString(message);
    UART_NextLine();
    UART_OutString(message);

    if(command[7] == '?'){
      BLT_InString(message);
      UART_NextLine();
      UART_OutString(message);
    }

    UART_NextLine();
    UART_NextLine();
  }

}
```

This bit of code above is the main function for setting up the HC-05 Bluetooth module. We first print out a welcome message with and example of a ‘command’. Then we wait for the user to input a command into the serial terminal, after receiving the command we then send it out to the Bluetooth module. After sending the command, we then wait for a response from the Bluetooth module, upon receiving the response we send that response to the serial terminal. Then we checked if the command has a question mark in it, if it does, we then wait for another response form the Bluetooth module. We also send this response to the serial terminal. This will loop till the user decides to turn off the microcontroller.

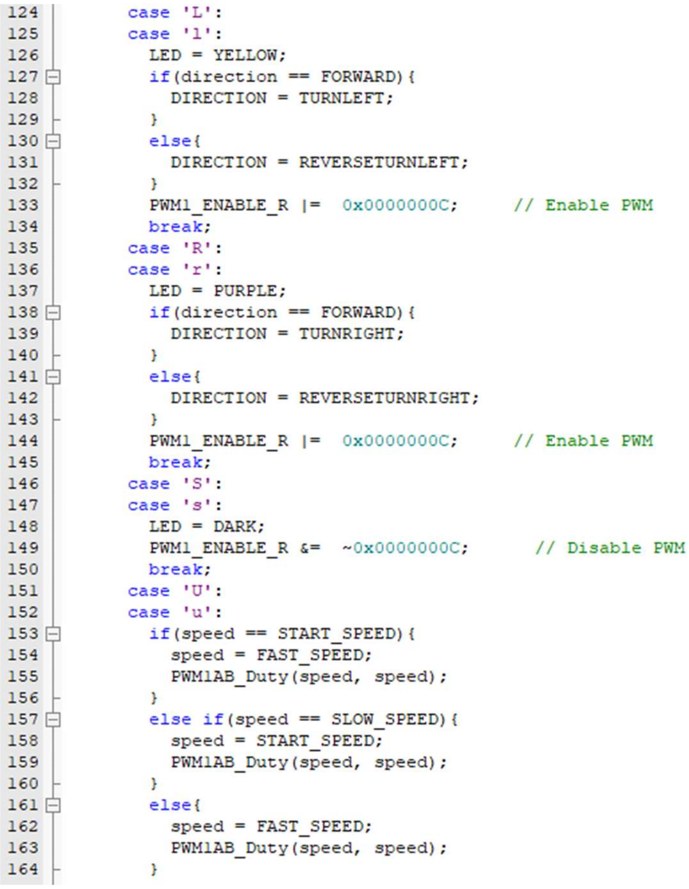


```

// PWM Constants
#define PERIOD          16000       // number of machine cycles for 1ms assuming 50MHz clock
#define START_SPEED     PERIOD*.6   // start speed for car
#define FAST_SPEED      PERIOD*.98  // Fastest speed of car
#define SLOW_SPEED      PERIOD*.3   // Slowest speed of car

// Constant declarations to access port registers using
// symbolic names instead of addresses
#define DIRECTION     (*((volatile unsigned long *)0x400040F0))
#define FORWARD     0x28
#define REVERSE     0x14
#define TURNLEFT    0x20
#define TURNRIGHT   0x08
#define REVERSETURNLEFT   0x04
#define REVERSETURNRIGHT  0x10

#define LED       (*((volatile unsigned long *)0x40025038))
// Color    LED(s) PortF
// dark     ---    0
// red      R--    0x02
// blue     --B    0x04
// green    -G-    0x08
// yellow   RG-    0x0A
// white    RGB    0x0E
// pink     R-B    0x06
// Cran     -GB    0x0C

#define DARK      0x00
#define RED       0x02
#define BLUE      0x04
#define GREEN     0x08
#define YELLOW    0x0A
#define CRAN      0x0C
#define WHITE     0x0E
#define PURPLE    0x06

```

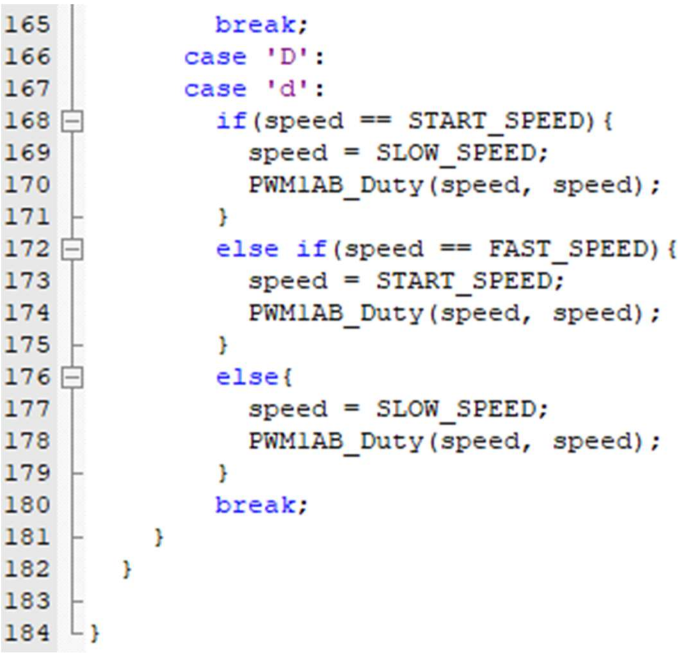
These are all the constant definitions we used to meet all the requirements of the remote-control part of the project. The PWM constants are used to control the speed of the robot car. The Direction constants are used to control what direction the robot car will turn/go. The LED constants are used to show the status of the robot car based on the color of the LED.



```c
int main(void){

  unsigned char control_symbol;
  unsigned int speed = START_SPEED;
  unsigned char direction = 0x00;

  UART_Init();
  BLT_Control_Init();
  PortF_LEDInit();
  PortA2_5_Init();
  PWM1G1AB_Init(PERIOD);

  // Initialize
  LED = DARK;
  PWM1AB_Duty(speed, speed);
  UART_OutString(">>> Welcome to Bluetooth Controlled Car App <<<");
  UART_NextLine();


  // Bluetooth Controlled Car
  while(1){
    control_symbol = BLT_InChar();
    UART_OutChar(control_symbol);
    UART_NextLine();

    switch(control_symbol){
      case 'F':
      case 'f':
        LED = GREEN;
        DIRECTION = FORWARD;
        direction = FORWARD;
        PWM1_ENABLE_R |=  0x0000000C;     // Enable PWM
        break;
      case 'B':
      case 'b':
        LED = BLUE;
        DIRECTION = REVERSE;
        direction = REVERSE;
        PWM1_ENABLE_R |=  0x0000000C;     // Enable PWM
        break;
```



```c
      case 'L':
      case 'l':
        LED = YELLOW;
        if(direction == FORWARD){
          DIRECTION = TURNLEFT;
        }
        else{
          DIRECTION = REVERSETURNLEFT;
        }
        PWM1_ENABLE_R |=  0x0000000C;     // Enable PWM
        break;
      case 'R':
      case 'r':
        LED = PURPLE;
        if(direction == FORWARD){
          DIRECTION = TURNRIGHT;
        }
        else{
          DIRECTION = REVERSETURNRIGHT;
        }
        PWM1_ENABLE_R |=  0x0000000C;     // Enable PWM
        break;
      case 'S':
      case 's':
        LED = DARK;
        PWM1_ENABLE_R &=  ~0x0000000C;      // Disable PWM
        break;
      case 'U':
      case 'u':
        if(speed == START_SPEED){
          speed = FAST_SPEED;
          PWM1AB_Duty(speed, speed);
        }
        else if(speed == SLOW_SPEED){
          speed = START_SPEED;
          PWM1AB_Duty(speed, speed);
        }
        else{
          speed = FAST_SPEED;
          PWM1AB_Duty(speed, speed);
        }
```



```
        break;
      case 'D':
      case 'd':
        if(speed == START_SPEED){
          speed = SLOW_SPEED;
          PWM1AB_Duty(speed, speed);
        }
        else if(speed == FAST_SPEED){
          speed = START_SPEED;
          PWM1AB_Duty(speed, speed);
        }
        else{
          speed = SLOW_SPEED;
          PWM1AB_Duty(speed, speed);
        }
        break;
    }
  }

}
```

The three screen shots above show the main function for the controlling of the robot car. First, we initialize all the Init function we need, and then initialize the LEDs to be off, set the speed of the robot car to be the start speed. Then when we enter the while loop, we first wait for a character from the Bluetooth module. Then we enter a switch statement and do the required function based on the character sent form the laptop/smartphone to the Bluetooth module and then to the microcontroller. The character and the associated functions are listed in Operation.

Conclusion

Implementing this Bluetooth Controlled Robot Car project was easier than we first anticipate. The code for the setting up of the HC-05 Bluetooth module was very easy because we were given an example project that has similar functionality of the code we wrote. The code for controlling the HC-05 Bluetooth module was also fairly easy, not as easy as the code for setting up the HC-05 Bluetooth module. The tricky part about coding for controlling the HC-05 Bluetooth module was trying to keep track of the all the different case statements in the switch. Other than that, it was a very straight forward and very helpful project. The example projects given were very helpful in the development of this project. Overall, this project was very helpful for us to understand and review topics like GPIO, hardware PWM, UART, power supply, HC-05 Bluetooth module, DC motors, and L298N motor drive controller.