CECS 447 - Spring 2024 - Project 3

Bluetooth Controlled Robot Car

By

Edward Ramos – Maryith Garzon - Rod Agatep (GROUP 9)

18 March 2024

Design a robot car and use a cell phone / PC Bluetooth terminal to control its movement.

Introduction

The purpose of this project is to design, code, and build a Bluetooth-controlled robot car. By integrating an HC-05 Bluetooth module, a microcontroller (such as the Launchpad), DC motors, and onboard LEDs, we aim to create a versatile and interactive mobile car robot. The robot will be wirelessly controlled using a Samsung smartphone or PC via Bluetooth communication.

Operation

Video Links:

Part 1 (Bluetooth Setup): https://youtu.be/D_DY8A6Fkpk

Part 2 Mode 1 (Demo): https://youtu.be/o1eF7EDbDPA

Part 2 Mode 2 (Free Driving): https://youtu.be/OGnm127XvM4

This program runs on a TM4C123G ARM Cortex Microcontroller; the code is written on Keil uVision5 and is downloaded to the board using the included USB cable.

The operation of this Bluetooth-controlled robot car project involves two main phases: setting up the HC-05 Bluetooth module and controlling the robot car via Bluetooth terminal application. In the setup phase, the HC-05 module is connected to the microcontroller using UART communication, allowing for the execution of configuration commands via a serial interface. The module is placed in command mode, where parameters such as Bluetooth name,

UART configuration, passcode, and role (Slave mode) are set. A setup program is developed to facilitate user interaction with the HC-05 module, allowing users to input AT commands through a serial terminal application. During this process, serial terminal input/output data is recorded to document the configuration steps.

Once the HC-05 module is configured, the robot car enters the operational phase, where it can be controlled wirelessly via a Bluetooth terminal application on a smartphone or PC. The car offers two modes of operation: Demo and Free Driving. In Demo mode, users can toggle between demonstrations such as drawing figures like an eight, circle, or square. In Free Driving mode, users have more direct control over the robot car's movements, including forward, reverse, left turn, right turn, stop, speed up, and slow down commands. LED indicators on the robot car denote its current mode, with green indicating Demo mode and blue indicating Free Driving mode. Overall, the operation of this project involves configuring the Bluetooth module for communication and implementing intuitive control mechanisms for the robot car's movements, providing an engaging and interactive experience for users.

The robot car will work on the following two modes:

1. Mode 1: Demo

When the robot car powers up, it automatically enters **Mode 1**. The green LED indicator will turn on. The robot car performs will perform specific demonstrations based on the commands received from the Bluetooth terminal app- Samsung smartphone. The car will perform a figure 8 (8), a circle (C), and a square (S).

2. Mode 2: Free Driving

In this mode, the user will use the bluettoth serial terminal and send commands to the car: forward (F), reverse (B), left turn (L), right turn (R), stop (S), speed up (U), and slow down (D).

Theory

TM4C Launchpad

The microcontroller platform that serves as the brain of the robot.

Romi Car

Equipped with two DC motors for movement.

L298N Motor Drive Controller (included in Romi Chassis)

Used to control the two DC motors.

Power Source (included in Romi Chassis)

At least 6 AA batteries to power the car and the LaunchPad.

HC-05 Bluetooth Module

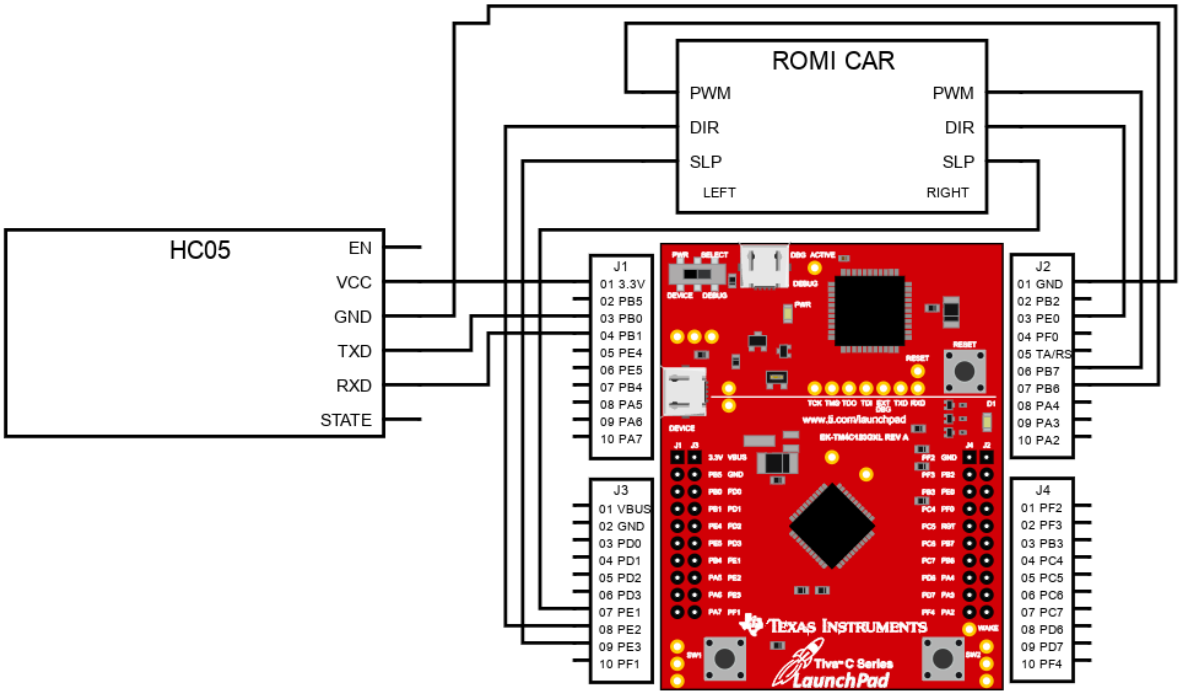
Enables wireless communication with external devices.

Tera Term

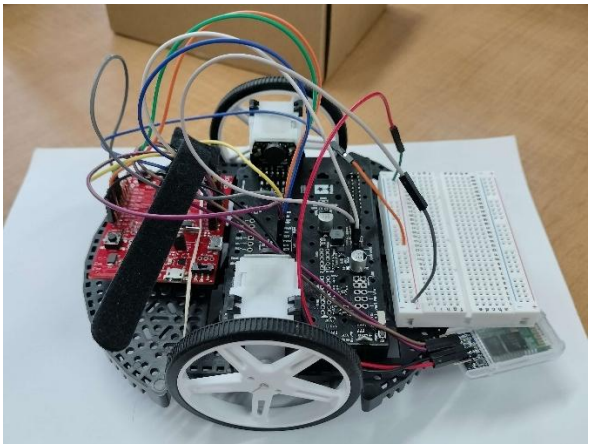
Serial Terminal Application. (used in part 1 of the project)

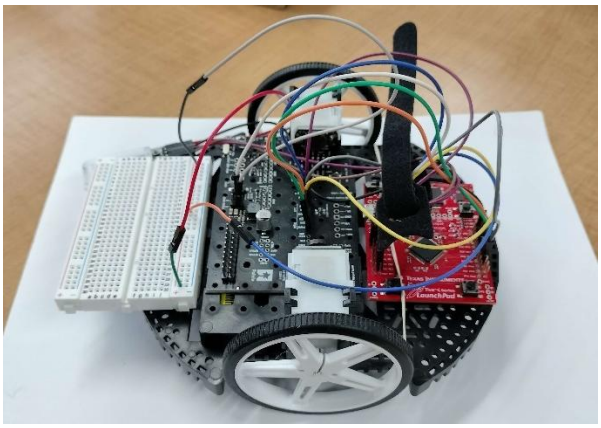
Hardware Design

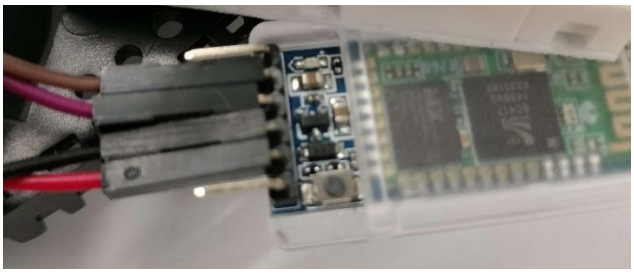
SCHEMATIC

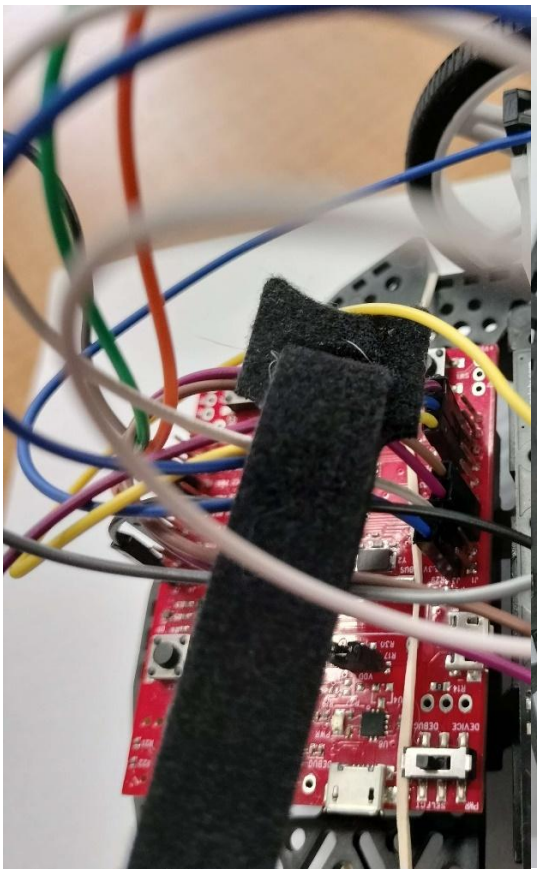


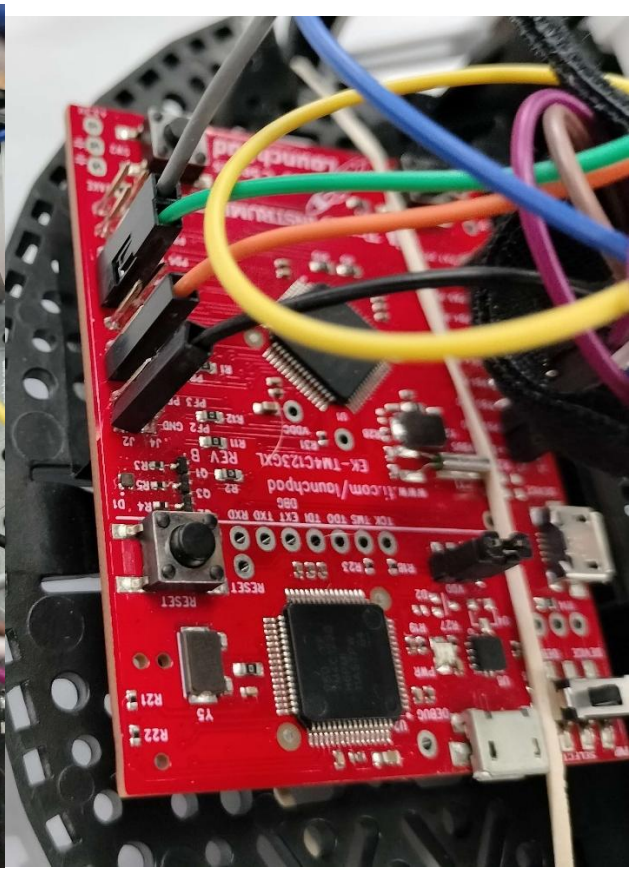
LAUNCHPAD AND CIRCUIT

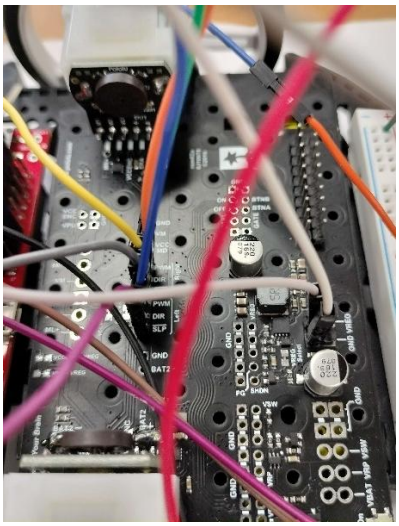




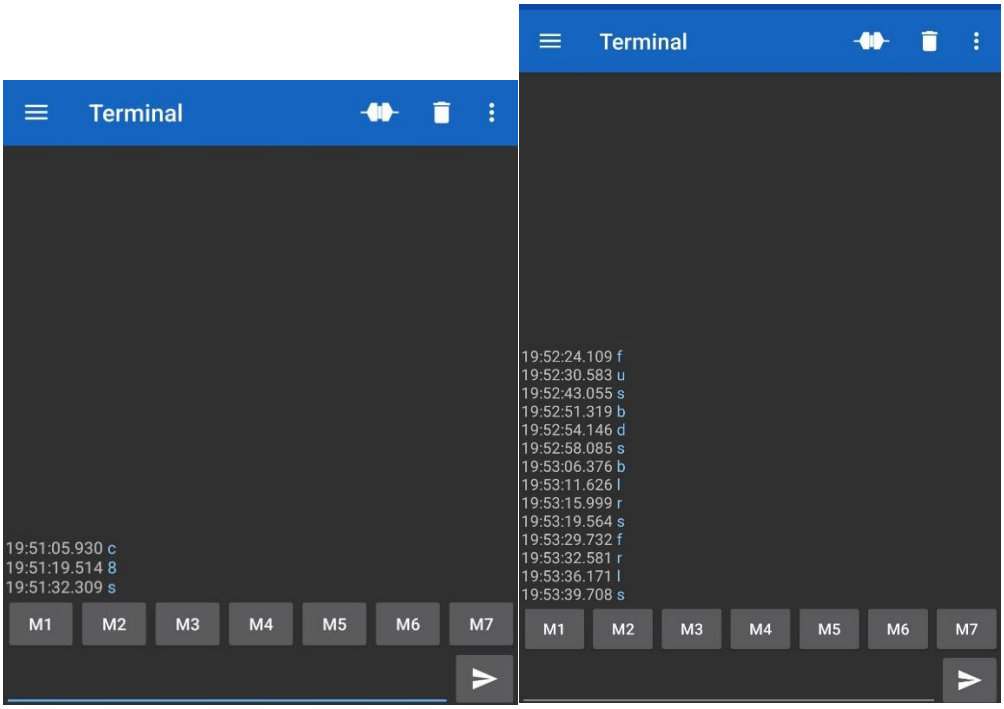








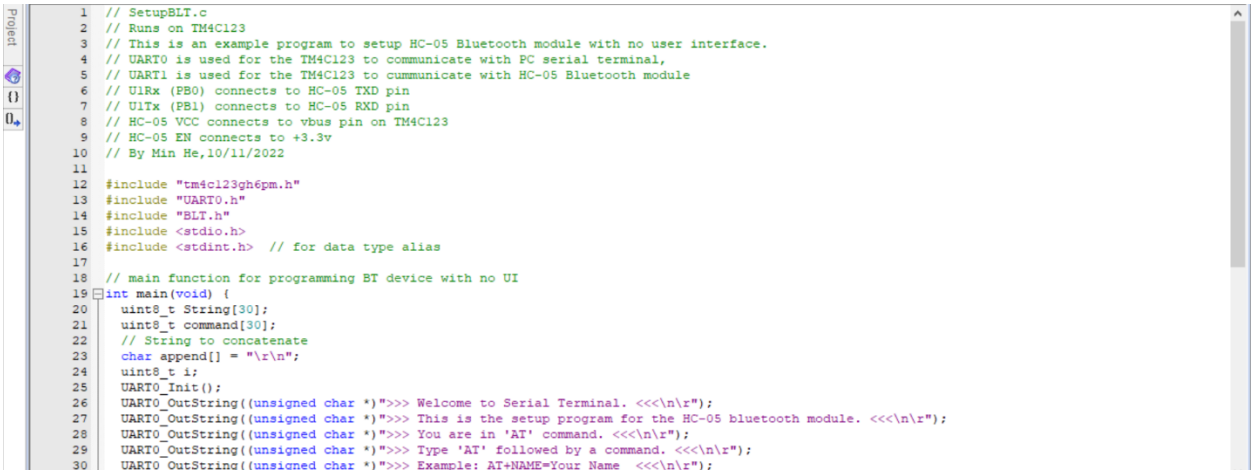
BLUETOOTH TERMINAL



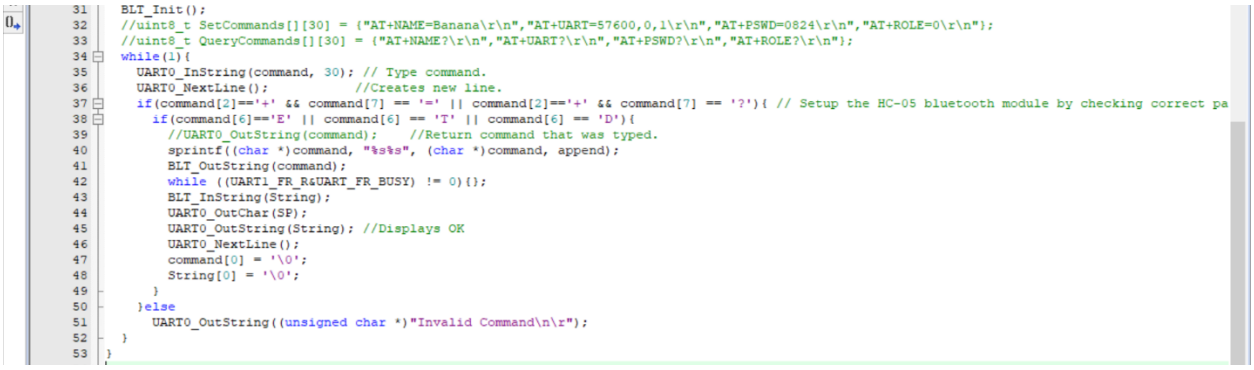
SOURCE CODE

Part 1 (Bluetooth Setup)

SetupBLT.c



```c
// SetupBLT.c
// Runs on TM4C123
// This is an example program to setup HC-05 Bluetooth module with no user interface.
// UART0 is used for the TM4C123 to communicate with PC serial terminal,
// UART1 is used for the TM4C123 to communicate with HC-05 Bluetooth module
// U1Rx (PB0) connects to HC-05 TXD pin
// U1Tx (PB1) connects to HC-05 RXD pin
// HC-05 VCC connects to vbus pin on TM4C123
// HC-05 EN connects to +3.3v
// By Min He,10/11/2022

#include "tm4c123gh6pm.h"
#include "UART0.h"
#include "BLT.h"
#include <stdio.h>
#include <stdint.h>  // for data type alias

// main function for programming BT device with no UI
int main(void) {
  uint8_t String[30];
  uint8_t command[30];
  // String to concatenate
  char append[] = "\r\n";
  uint8_t i;
  UART0_Init();
  UART0_OutString((unsigned char *)">>> Welcome to Serial Terminal. <<<\n\r");
  UART0_OutString((unsigned char *)">>> This is the setup program for the HC-05 bluetooth module. <<<\n\r");
  UART0_OutString((unsigned char *)">>> You are in 'AT' command. <<<\n\r");
  UART0_OutString((unsigned char *)">>> Type 'AT' followed by a command. <<<\n\r");
  UART0_OutString((unsigned char *)">>> Example: AT+NAME=Your Name  <<<\n\r");
```

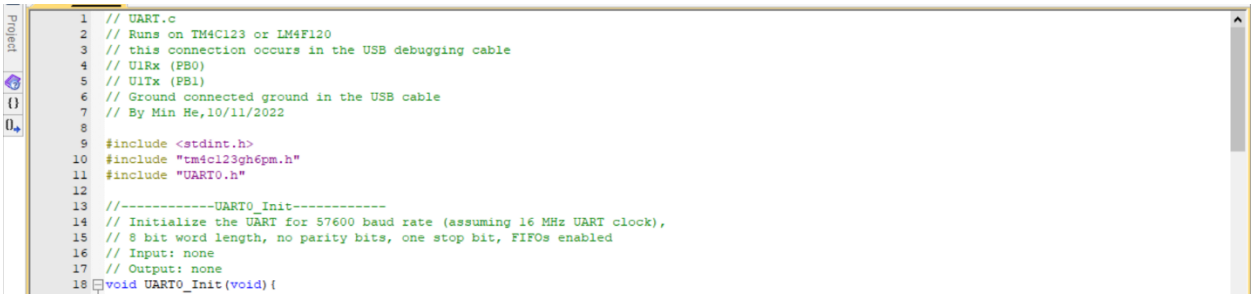


```c
  BLT_Init();
  //uint8_t SetCommands[][30] = {"AT+NAME=Banana\r\n","AT+UART=57600,0,1\r\n","AT+PSWD=0824\r\n","AT+ROLE=0\r\n"};
  //uint8_t QueryCommands[][30] = {"AT+NAME?\r\n","AT+UART?\r\n","AT+PSWD?\r\n","AT+ROLE?\r\n"};
  while(1){
    UART0_InString(command, 30); // Type command.
    UART0_NextLine();            //Creates new line.
    if(command[2]=='+' && command[7] == '=' || command[2]=='+' && command[7] == '?'){ // Setup the HC-05 bluetooth module by checking correct pa
      if(command[6]=='E' || command[6] == 'T' || command[6] == 'D'){
        //UART0_OutString(command);    //Return command that was typed.
        sprintf((char *)command, "%s%s", (char *)command, append);
        BLT_OutString(command);
        while ((UART1_FR_R&UART_FR_BUSY) != 0){};
        BLT_InString(String);
        UART0_OutChar(SP);
        UART0_OutString(String); //Displays OK
        UART0_NextLine();
        command[0] = '\0';
        String[0] = '\0';
      }
    }else
      UART0_OutString((unsigned char *)"Invalid Command\n\r");
  }
}
```

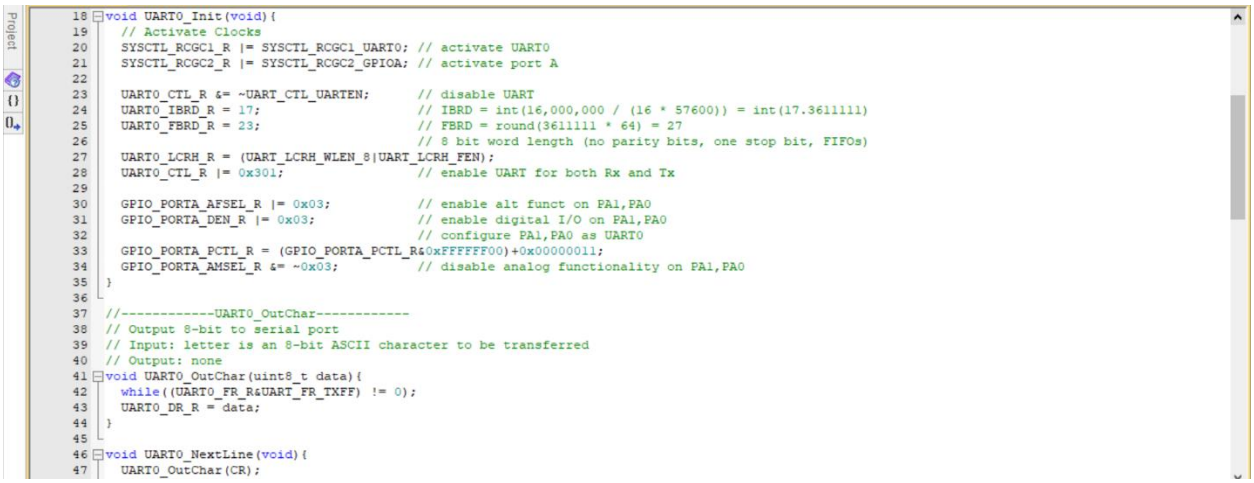
BLT.c

```c
// BLT.c
// Runs on TM4C123
// this connection occurs in the USB debugging cable
// U1Rx (PB0)
// U1Tx (PB1)
// Ground connected ground in the USB cable
// By Min He,10/11/2022

#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "BLT.h"
#include "UART0.h" // definition for LF

//------------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_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
}

//------------BLT_OutChar------------
// Output 8-bit to serial port UART1
// Input: letter is an 8-bit ASCII character to be transferred
// Output: none
void BLT_OutChar(uint8_t data){
  while((UART1_FR_R&UART_FR_TXFF) != 0);
  UART1_DR_R = data;
}

//------------BLT_OutString------------
// Output String (NULL termination)
// Input: pointer to a NULL-terminated string to be transferred
// Output: none
void BLT_OutString(uint8_t *pt){
  while(*pt){
    BLT_OutChar(*pt);
    pt++;
  }
}

//------------BLT_InChar------------
// Wait for new serial port input
// Input: none
// Output: ASCII code for key typed
uint8_t BLT_InChar(void){
  while((UART1_FR_R&UART_FR_RXFE) != 0);
  return((uint8_t)(UART1_DR_R&0xFF));
}


// This function reads response from HC-05 Bluetooth module.
// this function brings out a null terminated C string.
void BLT_InString(uint8_t *bufPt) {
  uint8_t length=0;

  // The follwoing code is based on the fact that a reply from HC-05 always end with "\r\n"
  do {
    bufPt[length] = BLT_InChar();
    length++;
  }while (bufPt[length-1]!=LF);

  // remove CR&LF and add null terminator
  length -= 2;
  bufPt[length] = 0;
}
```

UART0.c



```
// UART.c
// Runs on TM4C123 or LM4F120
// this connection occurs in the USB debugging cable
// U1Rx (PB0)
// U1Tx (PB1)
// Ground connected ground in the USB cable
// By Min He,10/11/2022

#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "UART0.h"

//------------UART0_Init------------
// Initialize the UART 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 UART0_Init(void){
```



```
void UART0_Init(void){
  // Activate Clocks
  SYSCTL_RCGC1_R |= SYSCTL_RCGC1_UART0; // activate UART0
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOA; // activate port A

  UART0_CTL_R &= ~UART_CTL_UARTEN;      // disable UART
  UART0_IBRD_R = 17;                    // IBRD = int(16,000,000 / (16 * 57600)) = int(17.3611111)
  UART0_FBRD_R = 23;                    // FBRD = round(3611111 * 64) = 27
                                        // 8 bit word length (no parity bits, one stop bit, FIFOs)
  UART0_LCRH_R = (UART_LCRH_WLEN_8|UART_LCRH_FEN);
  UART0_CTL_R |= 0x301;                 // enable UART for both Rx and Tx

  GPIO_PORTA_AFSEL_R |= 0x03;           // enable alt funct on PA1,PA0
  GPIO_PORTA_DEN_R |= 0x03;             // enable digital I/O on PA1,PA0
                                        // configure PA1,PA0 as UART0
  GPIO_PORTA_PCTL_R = (GPIO_PORTA_PCTL_R&0xFFFFFF00)+0x00000011;
  GPIO_PORTA_AMSEL_R &= ~0x03;          // disable analog functionality on PA1,PA0
}

//------------UART0_OutChar------------
// Output 8-bit to serial port
// Input: letter is an 8-bit ASCII character to be transferred
// Output: none
void UART0_OutChar(uint8_t data){
  while((UART0_FR_R&UART_FR_TXFF) != 0);
  UART0_DR_R = data;
}

void UART0_NextLine(void){
  UART0_OutChar(CR);
```

```
48    UART0_OutChar(LF);
49  }
50
51  //------------UART0_OutString------------
52  // Output String (NULL termination)
53  // Input: pointer to a NULL-terminated string to be transferred
54  // Output: none
55  void UART0_OutString(uint8_t *pt){
56    while(*pt){
57      UART0_OutChar(*pt);
58      pt++;
59    }
60  }
61
62  //------------UART0_InChar------------
63  // Wait for new serial port input
64  // Input: none
65  // Output: ASCII code for key typed
66  uint8_t UART0_InChar(void){
67    while((UART0_FR_R&UART_FR_RXFE) != 0);
68    return((uint8_t)(UART0_DR_R&0xFF));
69  }
70
71
72  //------------UART0_InString------------
73  // Accepts ASCII characters from the serial port
74  //    and adds them to a string until <enter> is typed
75  //    or until max length of the string is reached.
76  // It echoes each character as it is inputted.
77  // If a backspace is inputted, the string is modified
```

```
78  //    and the backspace is echoed
79  // terminates the string with a null character
80  // uses busy-waiting synchronization on RDRF
81  // Input: pointer to empty buffer, size of buffer
82  // Output: Null terminated string
83  // -- Modified by Agustinus Darmawan + Mingjie Qiu --
84  uint16_t UART0_InString(uint8_t *bufPt, uint16_t max) {
85  uint16_t length=0;
86  uint8_t character;
87    character = UART0_InChar();
88    while(character != CR){
89      if(character == BS){
90        if(length){
91          bufPt--;
92          length--;
93          UART0_OutChar(BS);
94        }
95      }
96      else if(length < max){
97        *bufPt = character;
98        bufPt++;
99        length++;
100       UART0_OutChar(character);
101     }
102     character = UART0_InChar();
103   }
104   *bufPt = 0;
105   return length;
106 }
```

Part 2 (Romi Car)

PWM.c

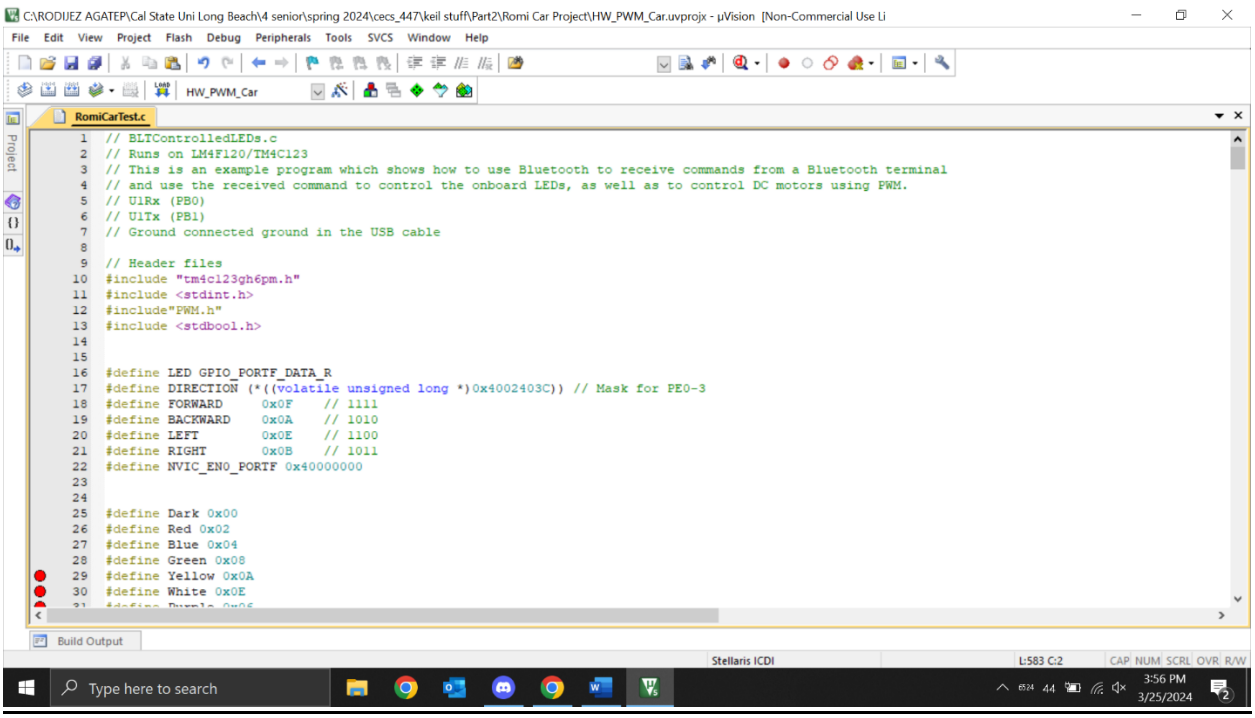
```
1   // PWM.c
2   // Runs on TM4C123
3   // Use PWM0/PB6 and PWM1/PB7 to generate pulse-width modulated outputs.
4   // Daniel Valvano
5   // March 28, 2014
6   // Modified by Min He, September 7, 2021
7
8   /* This example accompanies the book
9      "Embedded Systems: Real Time Interfacing to ARM Cortex M Microcontrollers",
10     ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2014
11    Program 6.7, section 6.3.2
12
13    Copyright 2014 by Jonathan W. Valvano, valvano@mail.utexas.edu
14       You may use, edit, run or distribute this file
15       as long as the above copyright notice remains
16    THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
17    OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
18    MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
19    VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
20    OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
21    For more information about my classes, my research, and my books, see
22    http://users.ece.utexas.edu/~valvano/
23    */
24  #include <stdint.h>
25  #include "tm4c123gh6pm.h"
26
27  #define DIRECTION (*((volatile unsigned long *)0x4002403C))
```

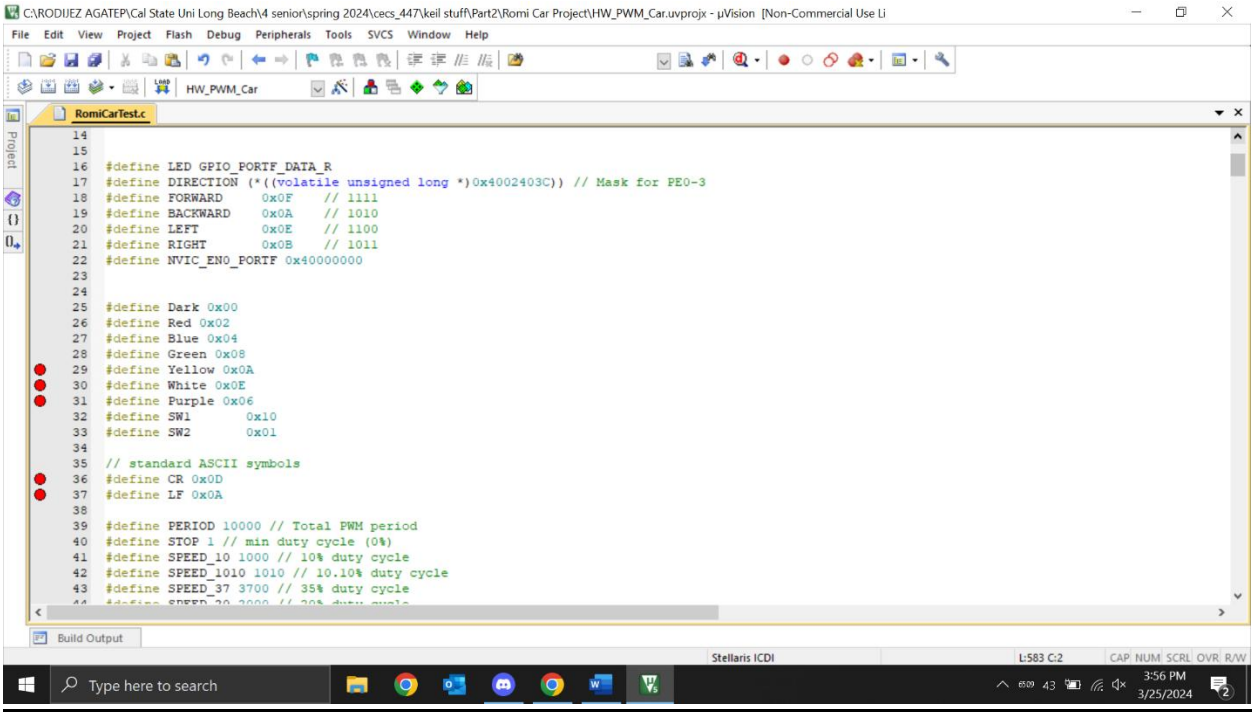
```
#define FORWARD     0x0F    // 1111
#define BACKWARD    0x0A    // 1010
#define LEFT        0x0E    // 1100
#define RIGHT       0x0B    // 1011

#define LIGHT (*((volatile unsigned long *)0x40025038))
#define RED 0x02
#define GREEN 0x08
#define BLUE 0x04

#define STOP 1
#define SPEED_35 3500
#define SPEED_60 6000
#define SPEED_80 8000
#define SPEED_98 9800

uint16_t DUTY = STOP;

// period is 16-bit number of PWM clock cycles in one period
// Output on PB6/M0PWM0
void PWM0A_Init(uint16_t period){
  SYSCTL_RCGCPWM_R |= 0x01;             // 1) activate PWM0
  SYSCTL_RCGCGPIO_R |= 0x02;            // 2) activate port B: 000010
  while((SYSCTL_RCGCGPIO_R&0x02) == 0){};
  GPIO_PORTB_AFSEL_R |= 0x40;           // enable alt funct on PB6: 0100 0000
  GPIO_PORTB_PCTL_R &= ~0x0F000000;     // configure PB6 as PWM0
  GPIO_PORTB_PCTL_R |= 0x04000000;
  GPIO_PORTB_AMSEL_R &= ~0x40;          // disable analog functionality on PB6
  GPIO_PORTB_DEN_R |= 0x40;             // enable digital I/O on PB6
  GPIO_PORTB_DR8R_R |= 0xC0;    // enable 8 mA drive on PB6,7
```

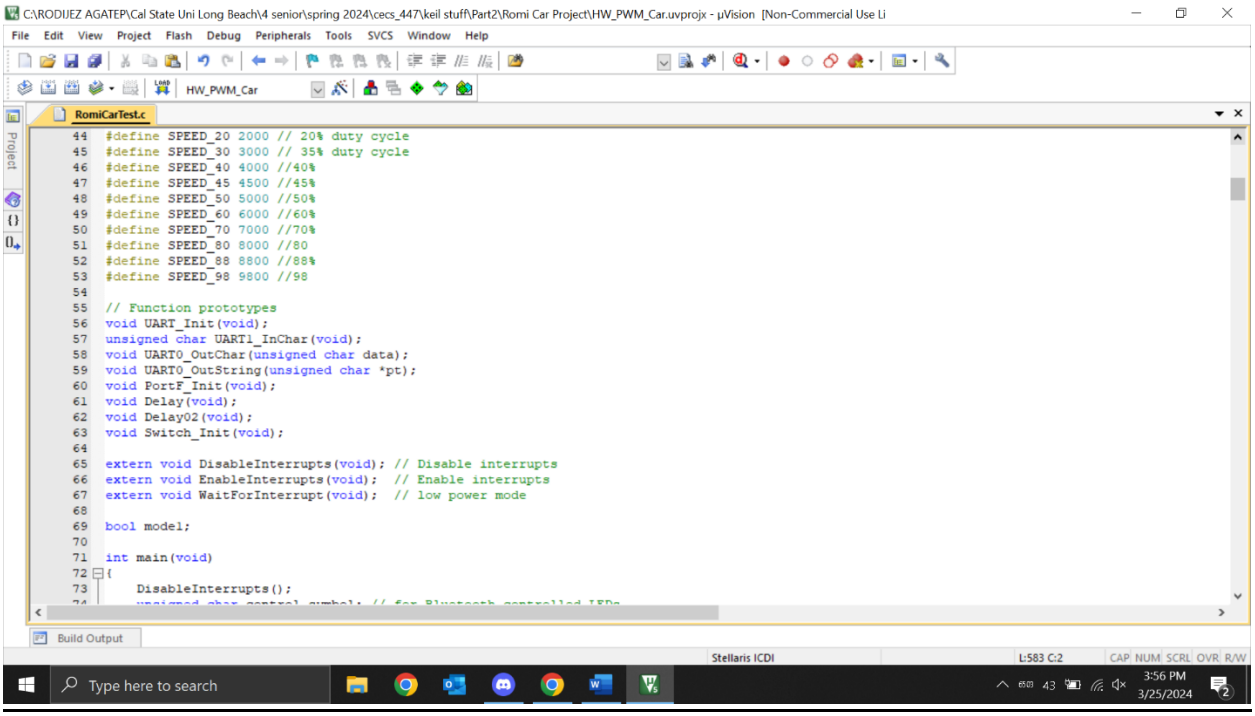
```
  SYSCTL_RCC_R = 0x00100000 |           // 3) use PWM divider
    (SYSCTL_RCC_R & (~0x001E0000));   //    configure for /2 divider: PWM clock: 80Mhz/2=40MHz
  PWM0_0_CTL_R = 0;                     // 4) re-loading down-counting mode
  PWM0_0_GENA_R = 0xC8;                 // low on LOAD, high on CMPA down
  // PB6 goes low on LOAD
  // PB6 goes high on CMPA down
  PWM0_0_LOAD_R = period - 1;           // 5) cycles needed to count down to 0
  PWM0_0_CMPA_R = 0;             // 6) count value when output rises
  PWM0_0_CTL_R |= 0x00000001;           // 7) start PWM0
  PWM0_ENABLE_R |= 0x00000001;          // enable PB6/M0PWM0
}
// change duty cycle of PB6
// duty is number of PWM clock cycles output is high
void PWM0A_Duty(uint16_t duty){
  PWM0_0_CMPA_R = duty - 1;             // 6) count value when output rises
}
// period is 16-bit number of PWM clock cycles in one period
// Output on PB7/M0PWM1
void PWM0B_Init(uint16_t period){
  volatile unsigned long delay;
  SYSCTL_RCGCPWM_R |= 0x01;             // 1) activate PWM0
  SYSCTL_RCGCGPIO_R |= 0x02;            // 2) activate port B
  delay = SYSCTL_RCGCGPIO_R;            // allow time to finish activating
  GPIO_PORTB_AFSEL_R |= 0x80;           // enable alt funct on PB7
  GPIO_PORTB_PCTL_R &= ~0xF0000000;     // configure PB7 as M0PWM1
  GPIO_PORTB_PCTL_R |= 0x40000000;
  GPIO_PORTB_AMSEL_R &= ~0x80;          // disable analog functionality on PB7
  GPIO_PORTB_DEN_R |= 0x80;             // enable digital I/O on PB7
  SYSCTL_RCC_R |= SYSCTL_RCC_USEPWMDIV; // 3) use PWM divider
  SYSCTL_RCC_R &= ~SYSCTL_RCC_PWMDIV_M; //    clear PWM divider field
```

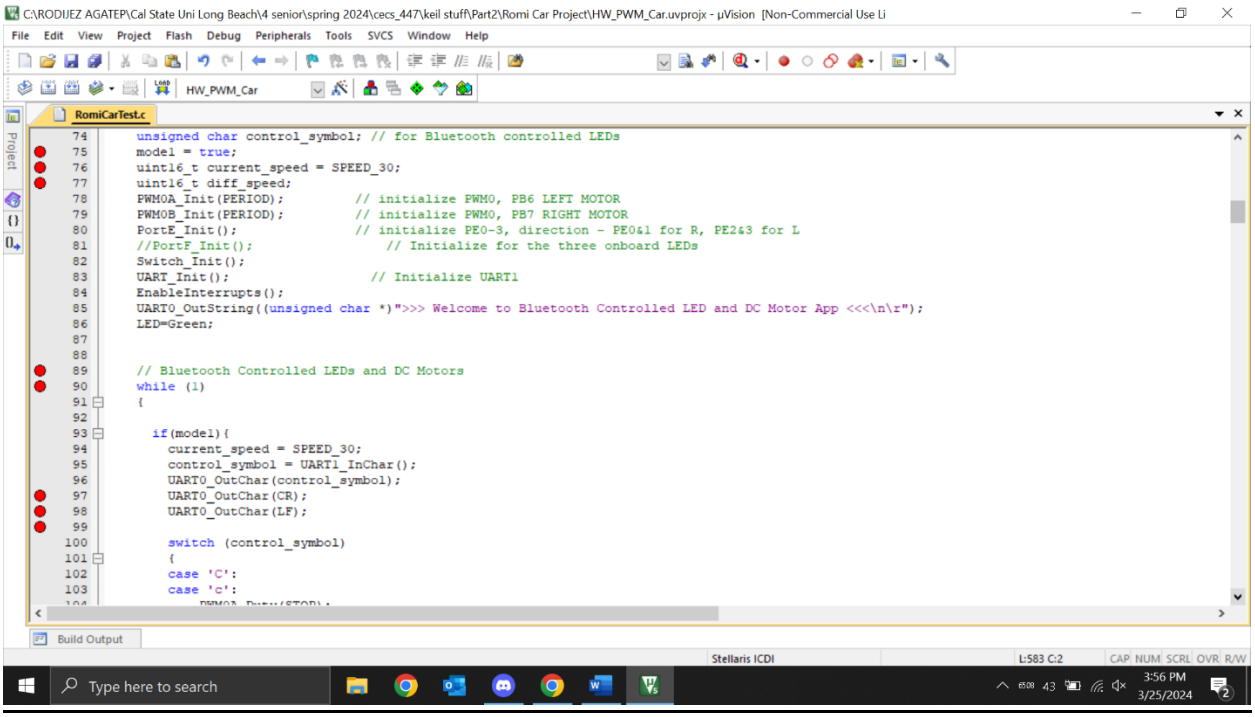
```
  SYSCTL_RCC_R += SYSCTL_RCC_PWMDIV_2;  //    configure for /2 divider
  PWM0_0_CTL_R = 0;                     // 4) re-loading down-counting mode
  PWM0_0_GENB_R = (PWM_0_GENB_ACTCMPBD_ONE|PWM_0_GENB_ACTLOAD_ZERO); // 0xC08
  // PB7 goes low on LOAD
  // PB7 goes high on CMPB down
  PWM0_0_LOAD_R = period - 1;           // 5) cycles needed to count down to 0: LAB2: PF1->M1PWM5:PWM1_2_
  PWM0_0_CMPB_R = 0;             // 6) count value when output rises: Lab 2:5%2=1->CMPB, GENB
  PWM0_0_CTL_R |= 0x00000001;           // 7) start PWM0: odd->B, even->A
  PWM0_ENABLE_R |= 0x00000002;          // enable PB7/M0PWM1
}
// change duty cycle of PB7
// duty is number of PWM clock cycles output is high
void PWM0B_Duty(uint16_t duty){
  PWM0_0_CMPB_R = duty - 1;             // 6) count value when output rises
}

// Subroutine to initialize port E pins PE0-3 for output
// PE0-3 control direction of motor
// Inputs: None
// Outputs: None
void PortE_Init(void){
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOE; //activate E clock
  while ((SYSCTL_RCGC2_R&SYSCTL_RCGC2_GPIOE)!= SYSCTL_RCGC2_GPIOE){} //wait for clk

  GPIO_PORTE_AMSEL_R &= ~0x0F; //disable analog function
  GPIO_PORTE_PCTL_R &= ~0x0000FFFF; //GPIO clear bit PCTL
  GPIO_PORTE_DIR_R |= 0x0F; //PE0-3 output
  GPIO_PORTE_AFSEL_R &= ~0x0F; //no alternate function
  GPIO_PORTE_DEN_R |= 0x0F; //enable digital pins PE0-3
}
```

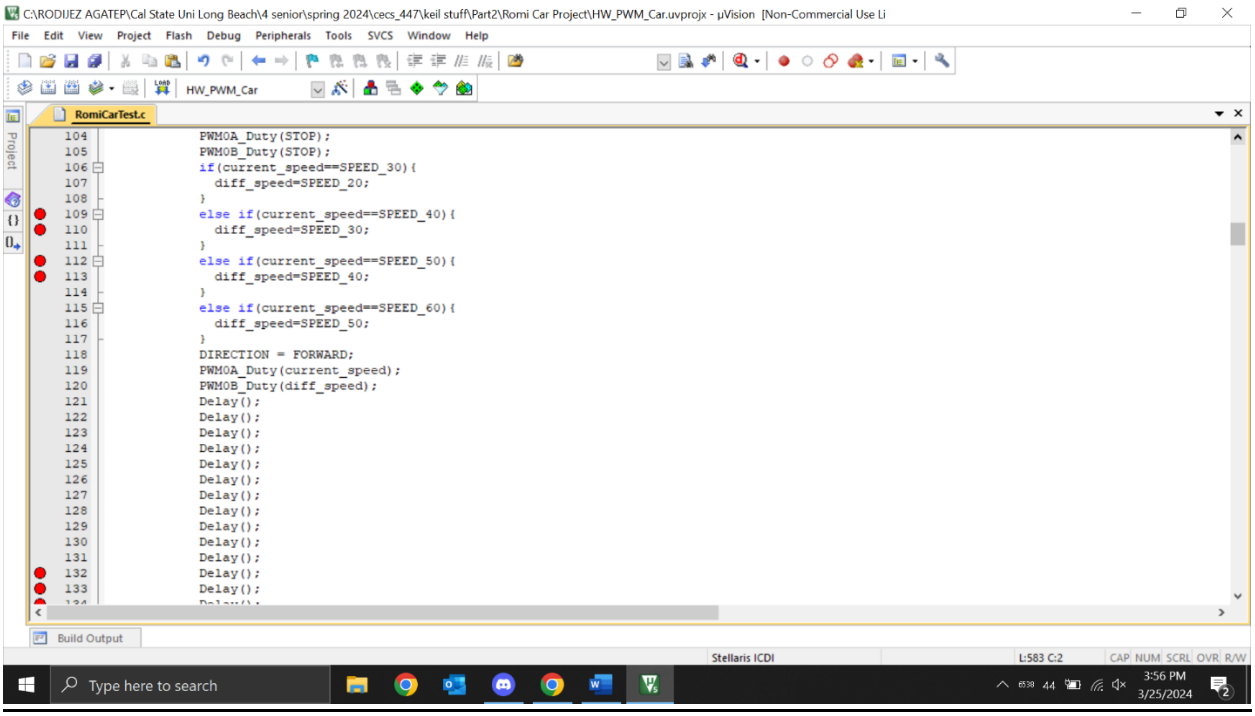
RomiCarTest.c

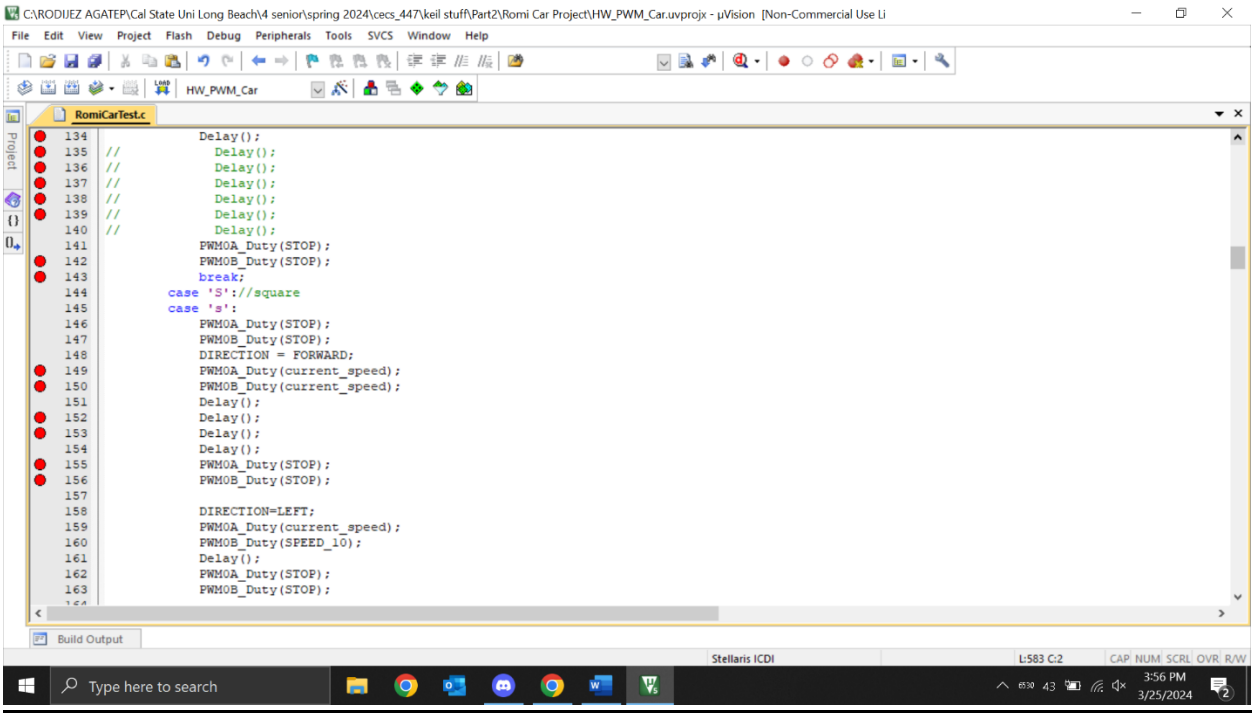


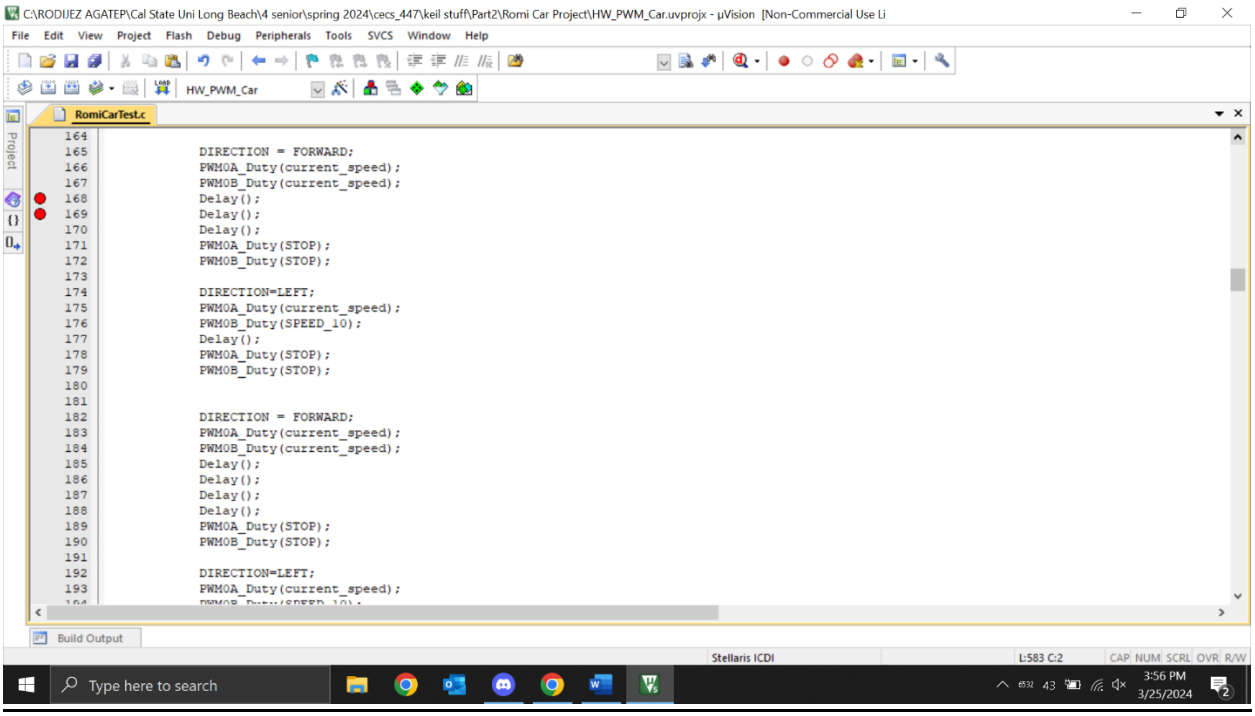


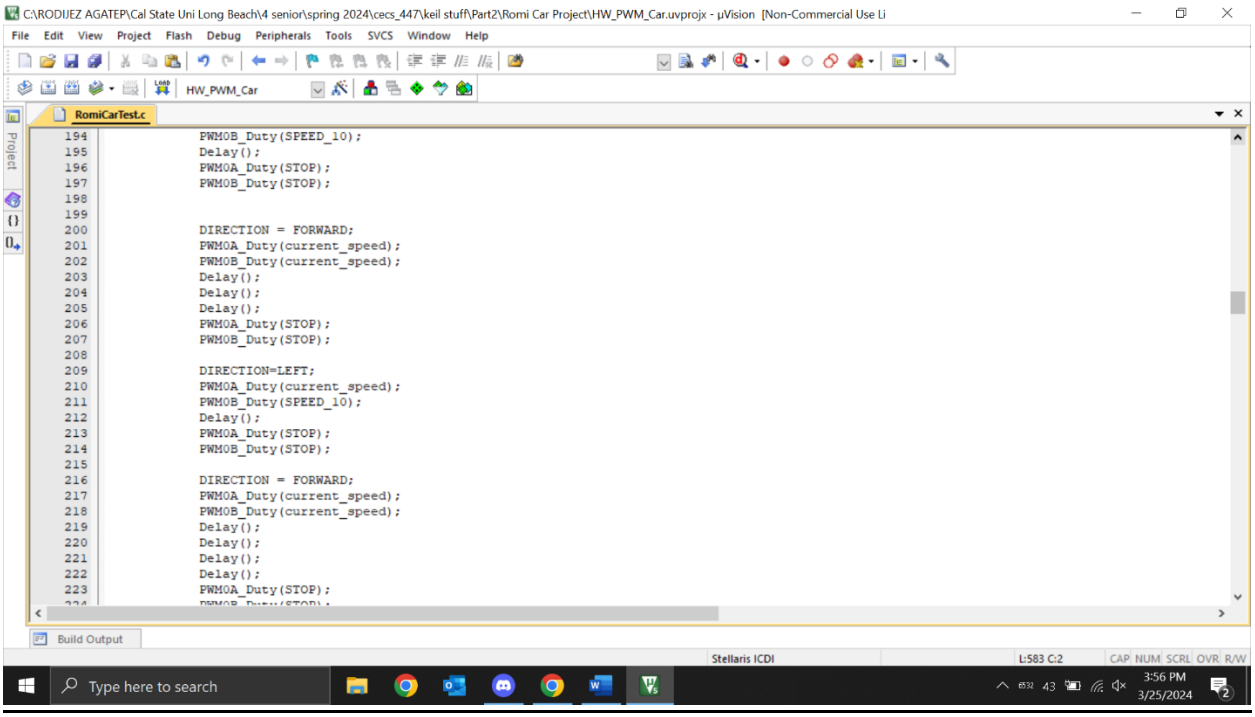

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
#define SPEED_20 2000 // 20% duty cycle
#define SPEED_30 3000 // 35% duty cycle
#define SPEED_40 4000 //40%
#define SPEED_45 4500 //45%
#define SPEED_50 5000 //50%
#define SPEED_60 6000 //60%
#define SPEED_70 7000 //70%
#define SPEED_80 8000 //80
#define SPEED_88 8800 //88%
#define SPEED_98 9800 //98
// Function prototypes
void UART_Init(void);
unsigned char UART1_InChar(void);
void UART0_OutChar(unsigned char data);
void UART0_OutString(unsigned char *pt);
void PortF_Init(void);
void Delay(void);
void Delay02(void);
void Switch_Init(void);
extern void DisableInterrupts(void); // Disable interrupts
extern void EnableInterrupts(void); // Enable interrupts
extern void WaitForInterrupt(void); // low power mode
bool model;
int main(void)
{
DisableInterrupts();
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:56 PM
3/25/2024

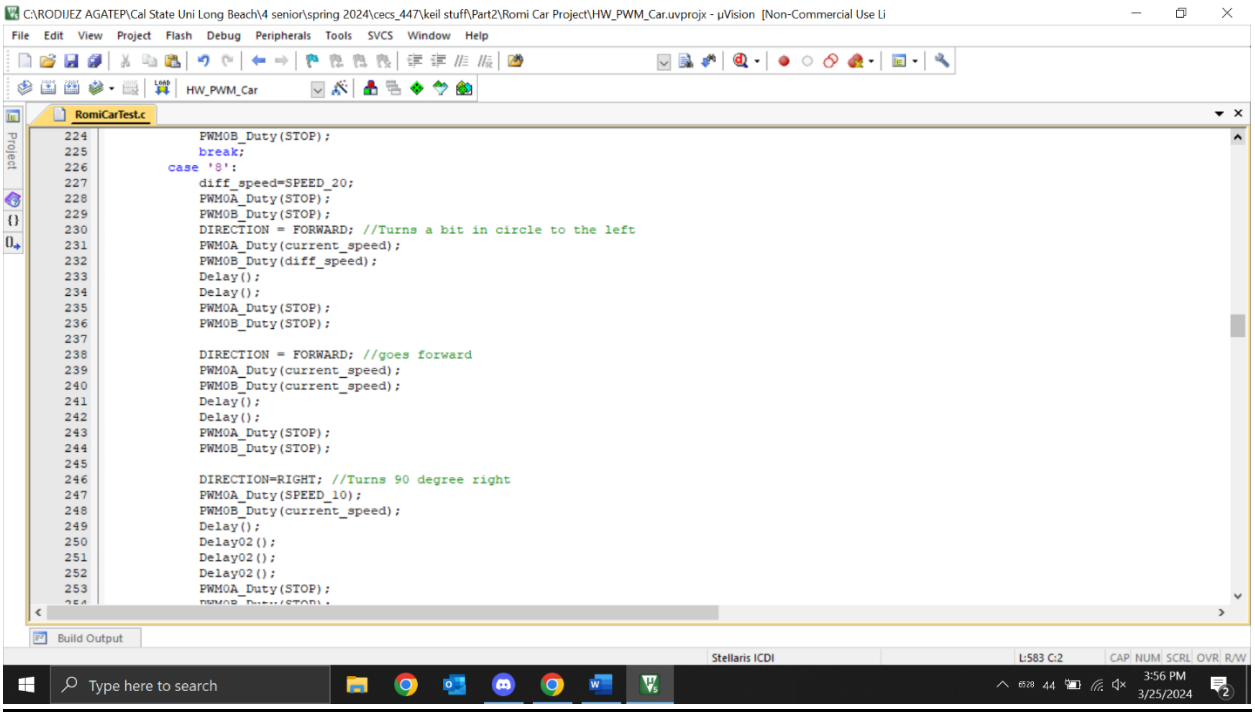

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
unsigned char control_symbol; // for Bluetooth controlled LEDs
model = true;
uint16_t current_speed = SPEED_30;
uint16_t diff_speed;
PWM0A_Init(PERIOD); // initialize PWM0, PB6 LEFT MOTOR
PWM0B_Init(PERIOD); // initialize PWM0, PB7 RIGHT MOTOR
PortE_Init(); // initialize PE0-3, direction - PE0&1 for R, PE2&3 for L
//PortF_Init(); // Initialize for the three onboard LEDs
Switch_Init();
UART_Init(); // Initialize UART1
EnableInterrupts();
UART0_OutString((unsigned char *)">>> Welcome to Bluetooth Controlled LED and DC Motor App <<<\n\r");
LED=Green;
// Bluetooth Controlled LEDs and DC Motors
while (1)
{
if(model){
current_speed = SPEED_30;
control_symbol = UART1_InChar();
UART0_OutChar(control_symbol);
UART0_OutChar(CR);
UART0_OutChar(LF);
switch (control_symbol)
{
case 'C':
case 'c':
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:56 PM
3/25/2024

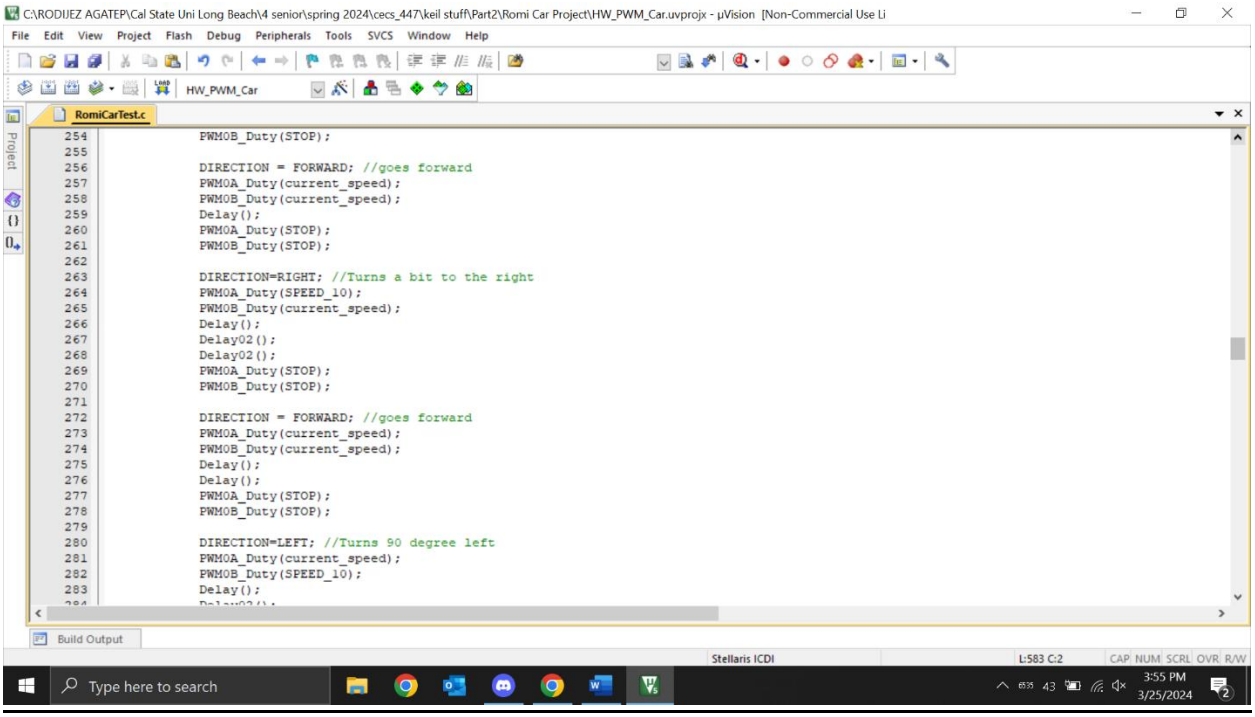

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
104 PWM0A_Duty(STOP);
105 PWM0B_Duty(STOP);
106 if(current_speed==SPEED_30){
107 diff_speed=SPEED_20;
108 }
109 else if(current_speed==SPEED_40){
110 diff_speed=SPEED_30;
111 }
112 else if(current_speed==SPEED_50){
113 diff_speed=SPEED_40;
114 }
115 else if(current_speed==SPEED_60){
116 diff_speed=SPEED_50;
117 }
118 DIRECTION = FORWARD;
119 PWM0A_Duty(current_speed);
120 PWM0B_Duty(diff_speed);
121 Delay();
122 Delay();
123 Delay();
124 Delay();
125 Delay();
126 Delay();
127 Delay();
128 Delay();
129 Delay();
130 Delay();
131 Delay();
132 Delay();
133 Delay();
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:56 PM
3/25/2024

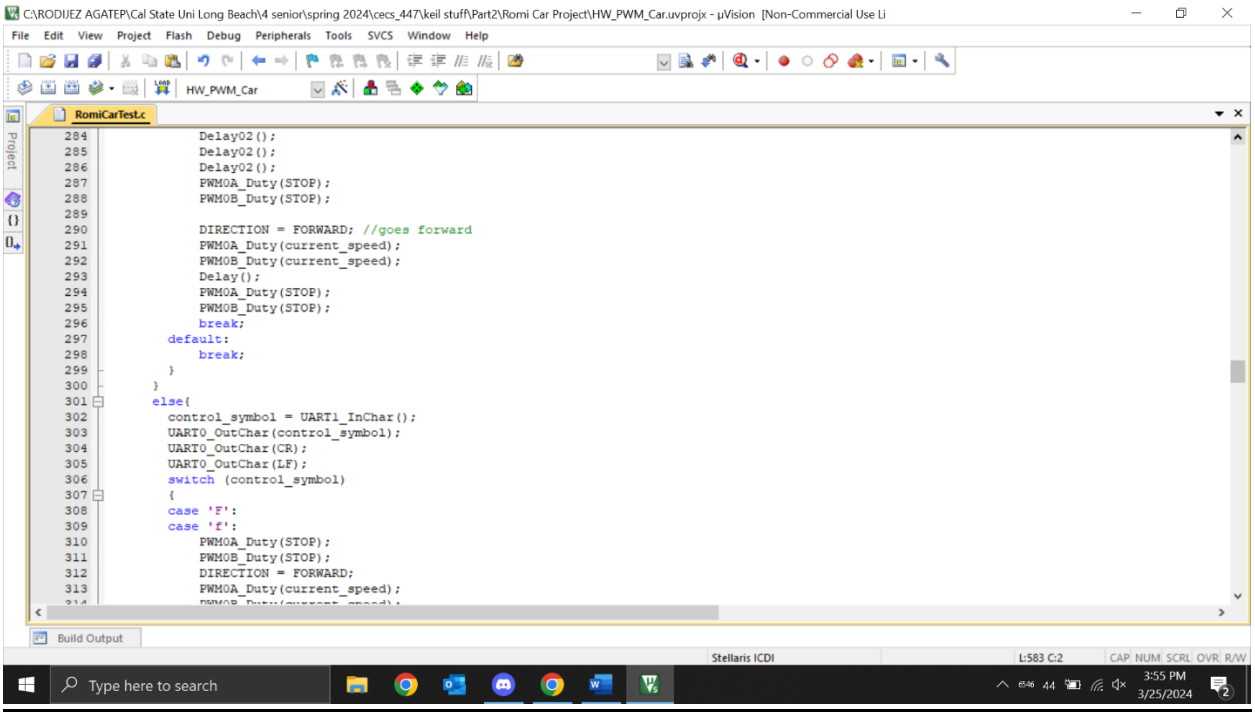

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
134 Delay();
135 // Delay();
136 // Delay();
137 // Delay();
138 // Delay();
139 // Delay();
140 // Delay();
141 PWM0A_Duty(STOP);
142 PWM0B_Duty(STOP);
143 break;
144 case 'S':square
145 case 's':
146 PWM0A_Duty(STOP);
147 PWM0B_Duty(STOP);
148 DIRECTION = FORWARD;
149 PWM0A_Duty(current_speed);
150 PWM0B_Duty(current_speed);
151 Delay();
152 Delay();
153 Delay();
154 Delay();
155 PWM0A_Duty(STOP);
156 PWM0B_Duty(STOP);
157
158 DIRECTION=LEFT;
159 PWM0A_Duty(current_speed);
160 PWM0B_Duty(SPEED_10);
161 Delay();
162 PWM0A_Duty(STOP);
163 PWM0B_Duty(STOP);
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:56 PM
3/25/2024

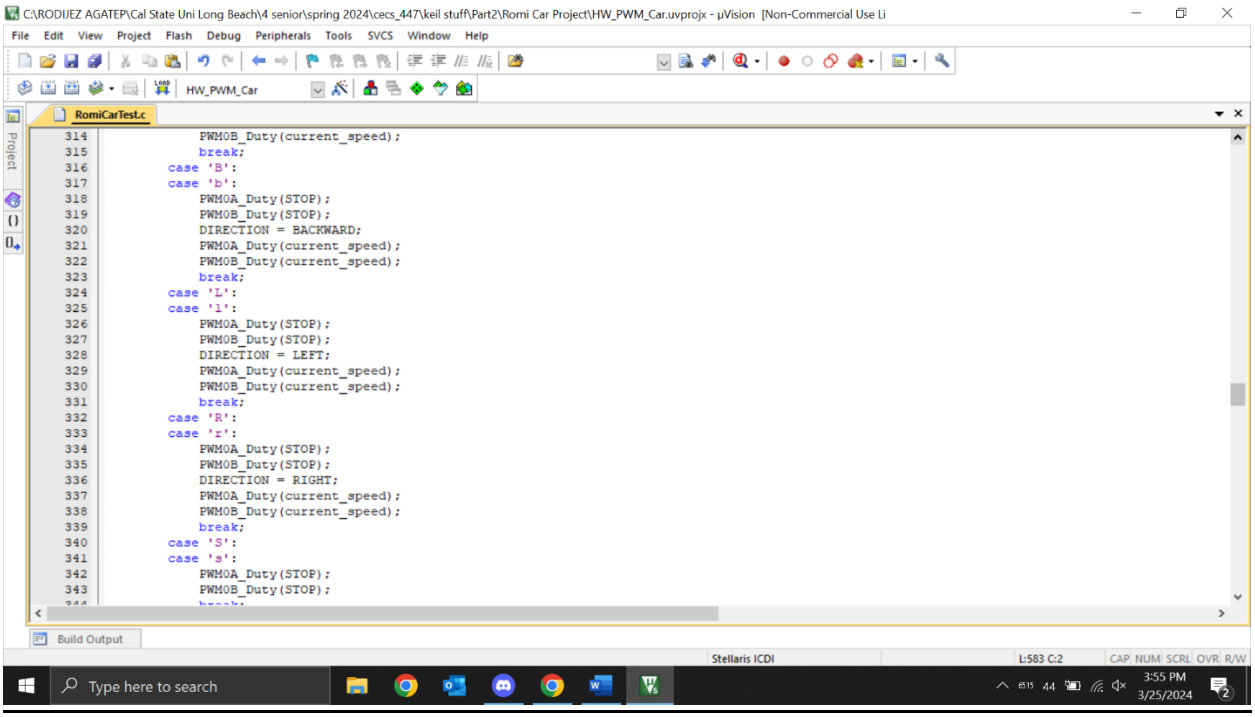

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
164 DIRECTION = FORWARD;
165 PWM0A_Duty(current_speed);
166 PWM0B_Duty(current_speed);
167 Delay();
168 Delay();
169 Delay();
170 PWM0A_Duty(STOP);
171 PWM0B_Duty(STOP);
172
173 DIRECTION=LEFT;
174 PWM0A_Duty(current_speed);
175 PWM0B_Duty(SPEED_10);
176 Delay();
177 PWM0A_Duty(STOP);
178 PWM0B_Duty(STOP);
179
180
181 DIRECTION = FORWARD;
182 PWM0A_Duty(current_speed);
183 PWM0B_Duty(current_speed);
184 Delay();
185 Delay();
186 Delay();
187 Delay();
188 PWM0A_Duty(STOP);
189 PWM0B_Duty(STOP);
190
191 DIRECTION=LEFT;
192 PWM0A_Duty(current_speed);
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:56 PM
3/25/2024

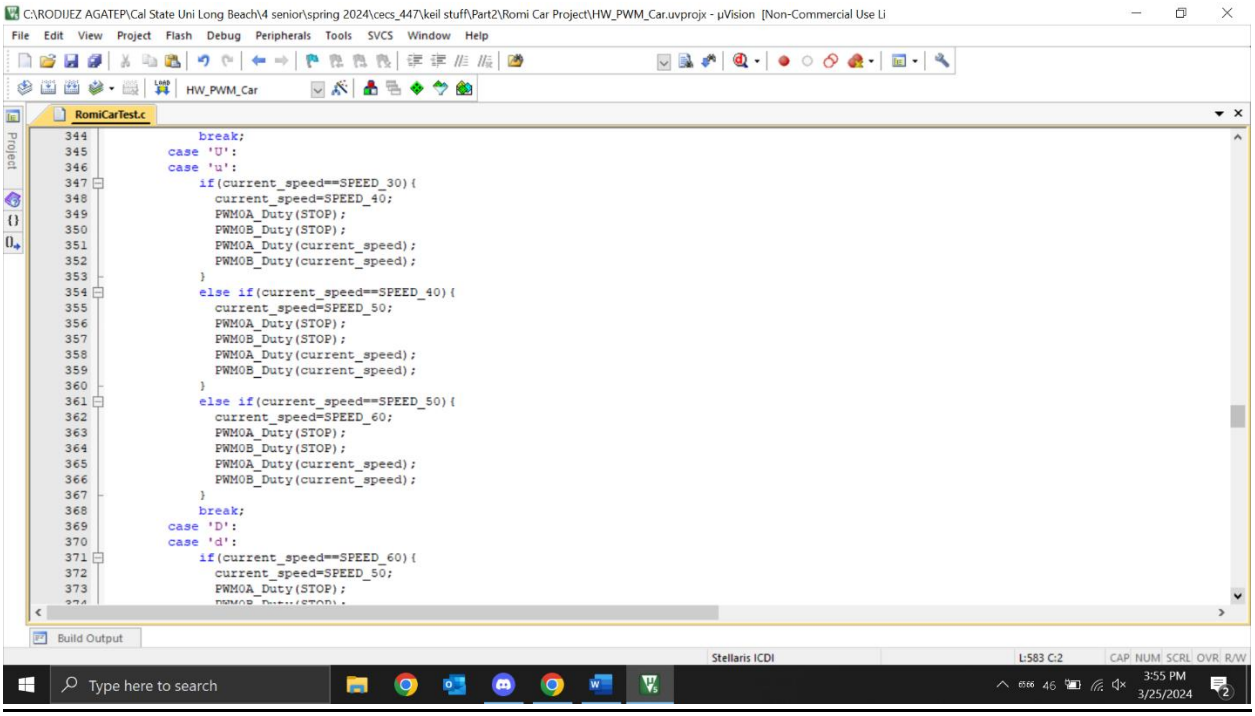

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
194 PWM0B_Duty(SPEED_10);
195 Delay();
196 PWM0A_Duty(STOP);
197 PWM0B_Duty(STOP);
198
199
200 DIRECTION = FORWARD;
201 PWM0A_Duty(current_speed);
202 PWM0B_Duty(current_speed);
203 Delay();
204 Delay();
205 Delay();
206 PWM0A_Duty(STOP);
207 PWM0B_Duty(STOP);
208
209 DIRECTION=LEFT;
210 PWM0A_Duty(current_speed);
211 PWM0B_Duty(SPEED_10);
212 Delay();
213 PWM0A_Duty(STOP);
214 PWM0B_Duty(STOP);
215
216 DIRECTION = FORWARD;
217 PWM0A_Duty(current_speed);
218 PWM0B_Duty(current_speed);
219 Delay();
220 Delay();
221 Delay();
222 Delay();
223 PWM0A_Duty(STOP);
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:56 PM
3/25/2024

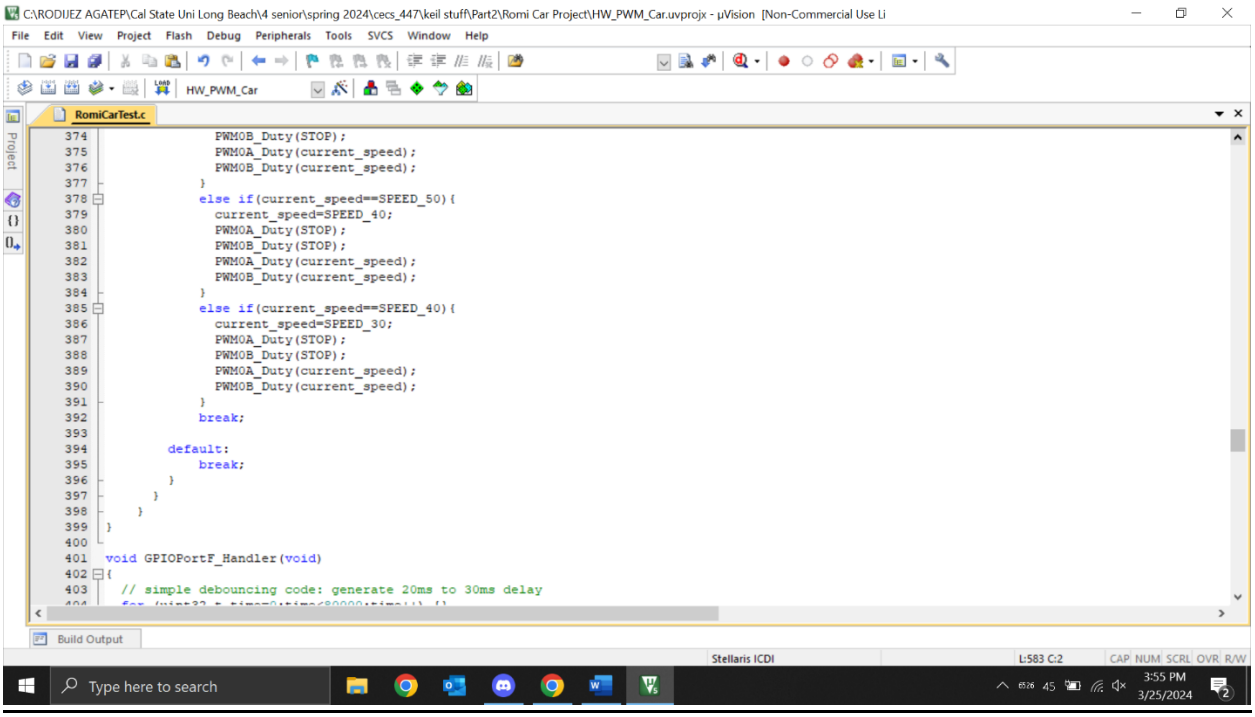

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
224 PWM0B_Duty(STOP);
225 break;
226 case '8':
227 diff_speed=SPEED_20;
228 PWM0A_Duty(STOP);
229 PWM0B_Duty(STOP);
230 DIRECTION = FORWARD; //Turns a bit in circle to the left
231 PWM0A_Duty(current_speed);
232 PWM0B_Duty(diff_speed);
233 Delay();
234 Delay();
235 PWM0A_Duty(STOP);
236 PWM0B_Duty(STOP);
237
238 DIRECTION = FORWARD; //goes forward
239 PWM0A_Duty(current_speed);
240 PWM0B_Duty(current_speed);
241 Delay();
242 Delay();
243 PWM0A_Duty(STOP);
244 PWM0B_Duty(STOP);
245
246 DIRECTION=RIGHT; //Turns 90 degree right
247 PWM0A_Duty(SPEED_10);
248 PWM0B_Duty(current_speed);
249 Delay();
250 Delay02();
251 Delay02();
252 Delay02();
253 PWM0A_Duty(STOP);
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:56 PM
3/25/2024

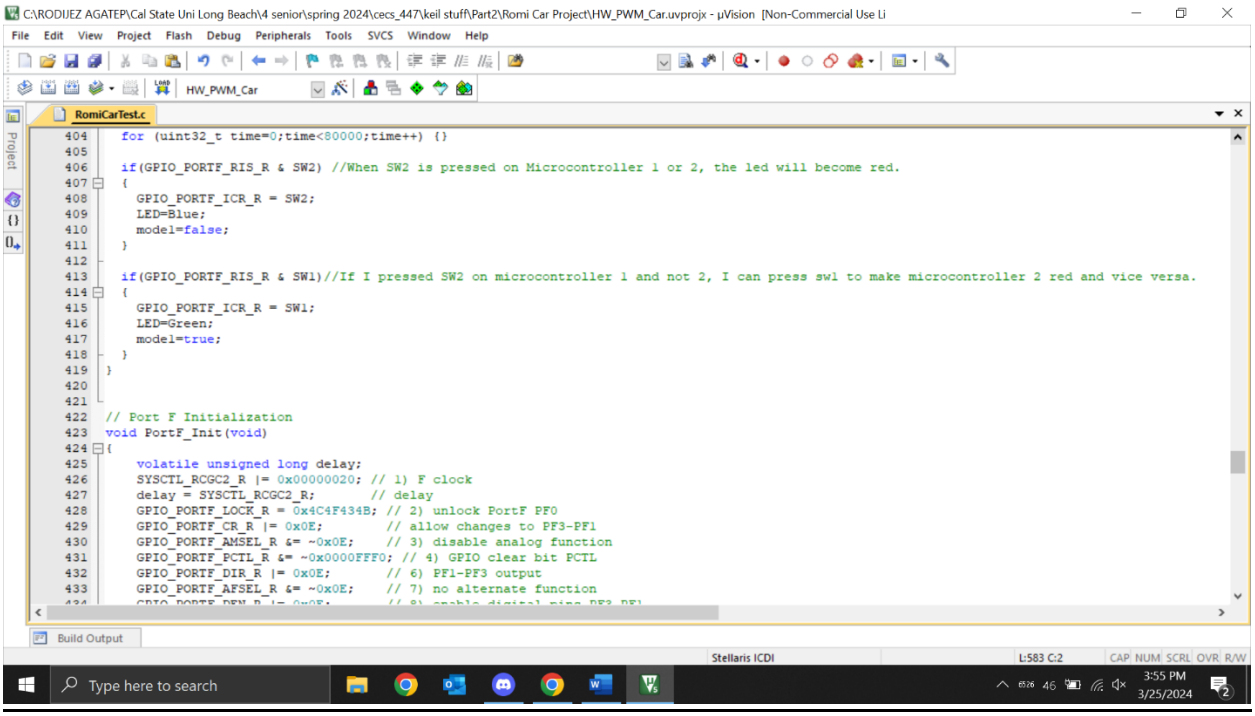

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
254 PWM0B_Duty(STOP);
255
256 DIRECTION = FORWARD; //goes forward
257 PWM0A_Duty(current_speed);
258 PWM0B_Duty(current_speed);
259 Delay();
260 PWM0A_Duty(STOP);
261 PWM0B_Duty(STOP);
262
263 DIRECTION=RIGHT; //Turns a bit to the right
264 PWM0A_Duty(SPEED_10);
265 PWM0B_Duty(current_speed);
266 Delay();
267 Delay02();
268 Delay02();
269 PWM0A_Duty(STOP);
270 PWM0B_Duty(STOP);
271
272 DIRECTION = FORWARD; //goes forward
273 PWM0A_Duty(current_speed);
274 PWM0B_Duty(current_speed);
275 Delay();
276 Delay();
277 PWM0A_Duty(STOP);
278 PWM0B_Duty(STOP);
279
280 DIRECTION=LEFT; //Turns 90 degree left
281 PWM0A_Duty(current_speed);
282 PWM0B_Duty(SPEED_10);
283 Delay();
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:55 PM
3/25/2024

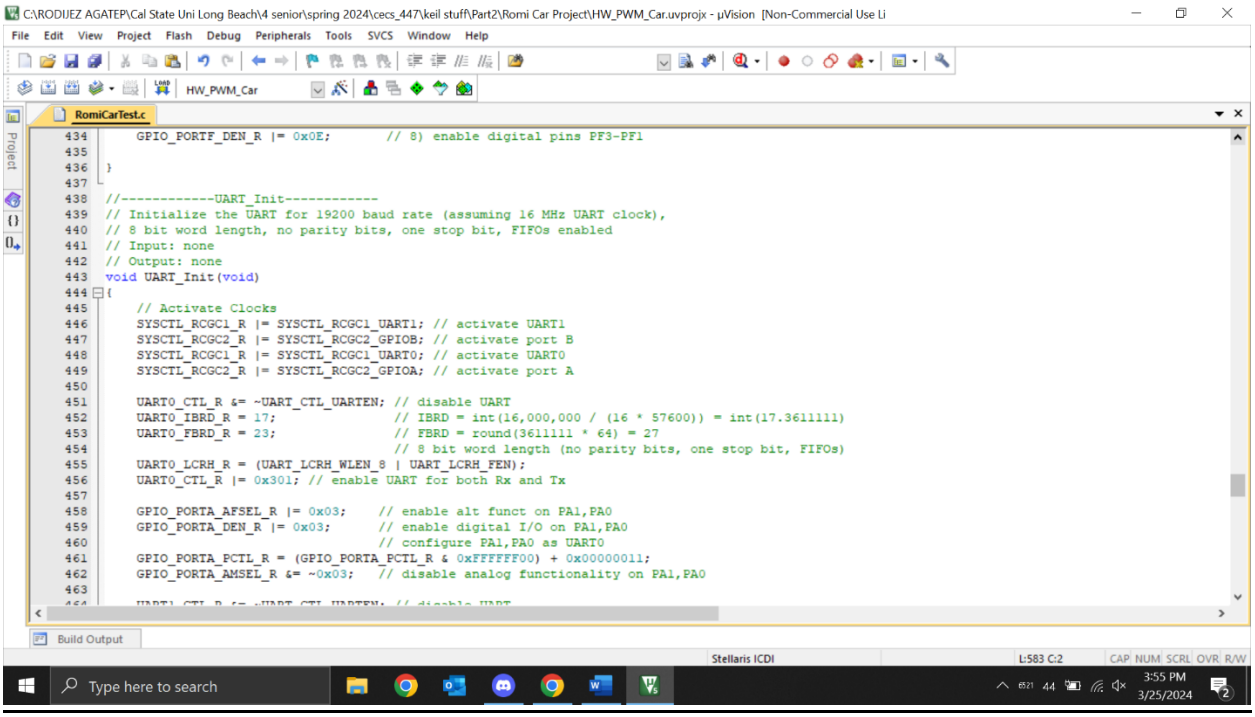

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
Delay02();
Delay02();
Delay02();
PWM0A_Duty(STOP);
PWM0B_Duty(STOP);
DIRECTION = FORWARD; //goes forward
PWM0A_Duty(current_speed);
PWM0B_Duty(current_speed);
Delay();
PWM0A_Duty(STOP);
PWM0B_Duty(STOP);
break;
default:
break;
}
}
else{
control_symbol = UART1_InChar();
UART0_OutChar(control_symbol);
UART0_OutChar(CR);
UART0_OutChar(LF);
switch (control_symbol)
{
case 'F':
case 'f':
PWM0A_Duty(STOP);
PWM0B_Duty(STOP);
DIRECTION = FORWARD;
PWM0A_Duty(current_speed);
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:55 PM
3/25/2024

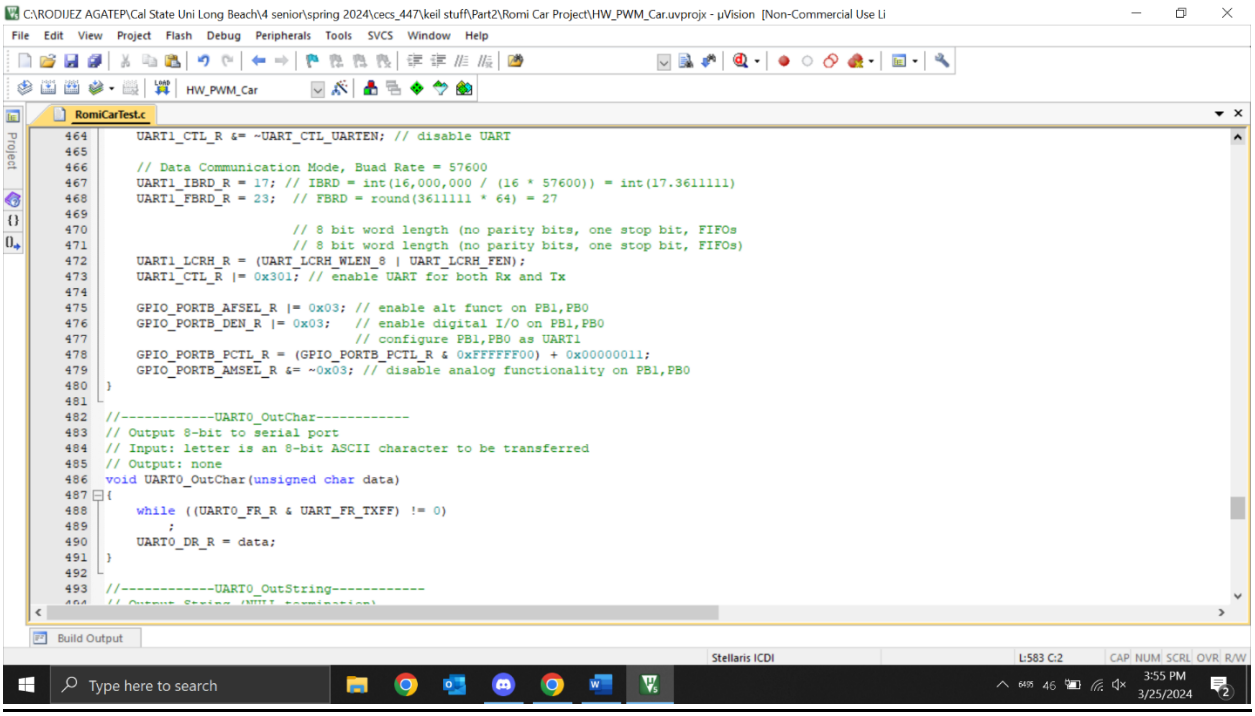

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
PWM0B_Duty(current_speed);
break;
case 'B':
case 'b':
PWM0A_Duty(STOP);
PWM0B_Duty(STOP);
DIRECTION = BACKWARD;
PWM0A_Duty(current_speed);
PWM0B_Duty(current_speed);
break;
case 'L':
case 'l':
PWM0A_Duty(STOP);
PWM0B_Duty(STOP);
DIRECTION = LEFT;
PWM0A_Duty(current_speed);
PWM0B_Duty(current_speed);
break;
case 'R':
case 'r':
PWM0A_Duty(STOP);
PWM0B_Duty(STOP);
DIRECTION = RIGHT;
PWM0A_Duty(current_speed);
PWM0B_Duty(current_speed);
break;
case 'S':
case 's':
PWM0A_Duty(STOP);
PWM0B_Duty(STOP);
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:55 PM
3/25/2024

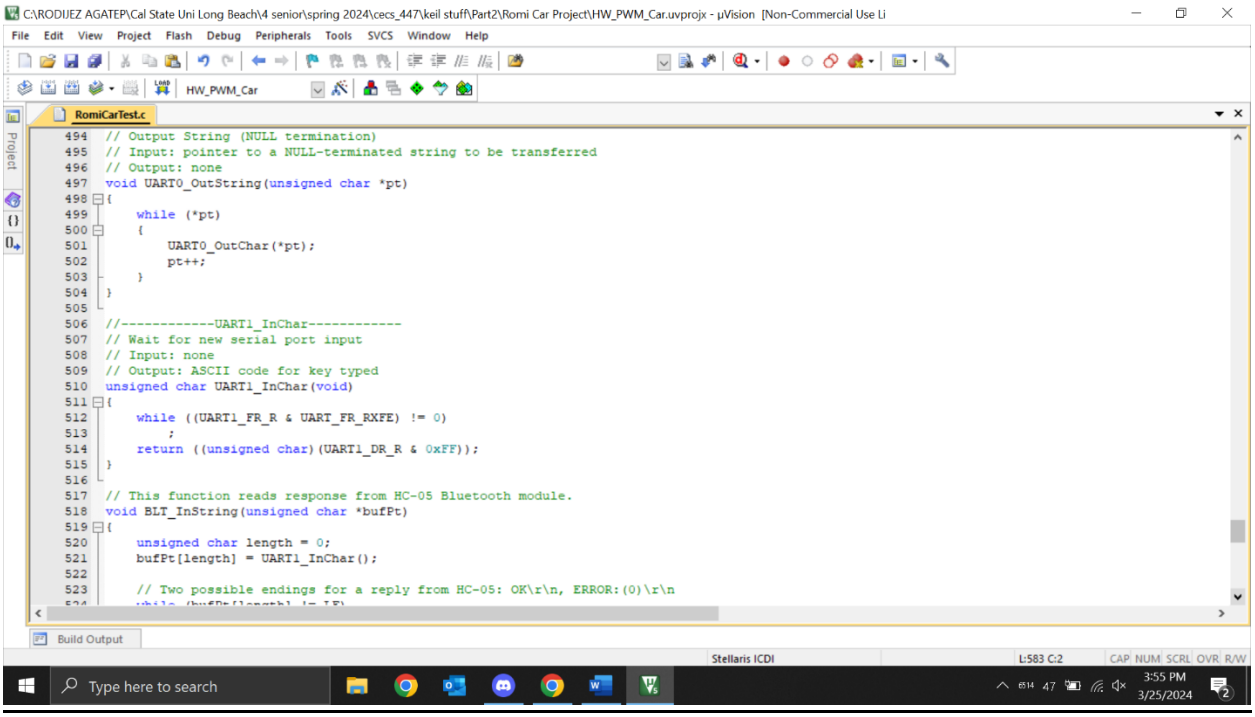

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
344 break;
345 case 'U':
346 case 'u':
347 if(current_speed==SPEED_30){
348 current_speed=SPEED_40;
349 PWM0A_Duty(STOP);
350 PWM0B_Duty(STOP);
351 PWM0A_Duty(current_speed);
352 PWM0B_Duty(current_speed);
353 }
354 else if(current_speed==SPEED_40){
355 current_speed=SPEED_50;
356 PWM0A_Duty(STOP);
357 PWM0B_Duty(STOP);
358 PWM0A_Duty(current_speed);
359 PWM0B_Duty(current_speed);
360 }
361 else if(current_speed==SPEED_50){
362 current_speed=SPEED_60;
363 PWM0A_Duty(STOP);
364 PWM0B_Duty(STOP);
365 PWM0A_Duty(current_speed);
366 PWM0B_Duty(current_speed);
367 }
368 break;
369 case 'D':
370 case 'd':
371 if(current_speed==SPEED_60){
372 current_speed=SPEED_50;
373 PWM0A_Duty(STOP);
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:55 PM
3/25/2024

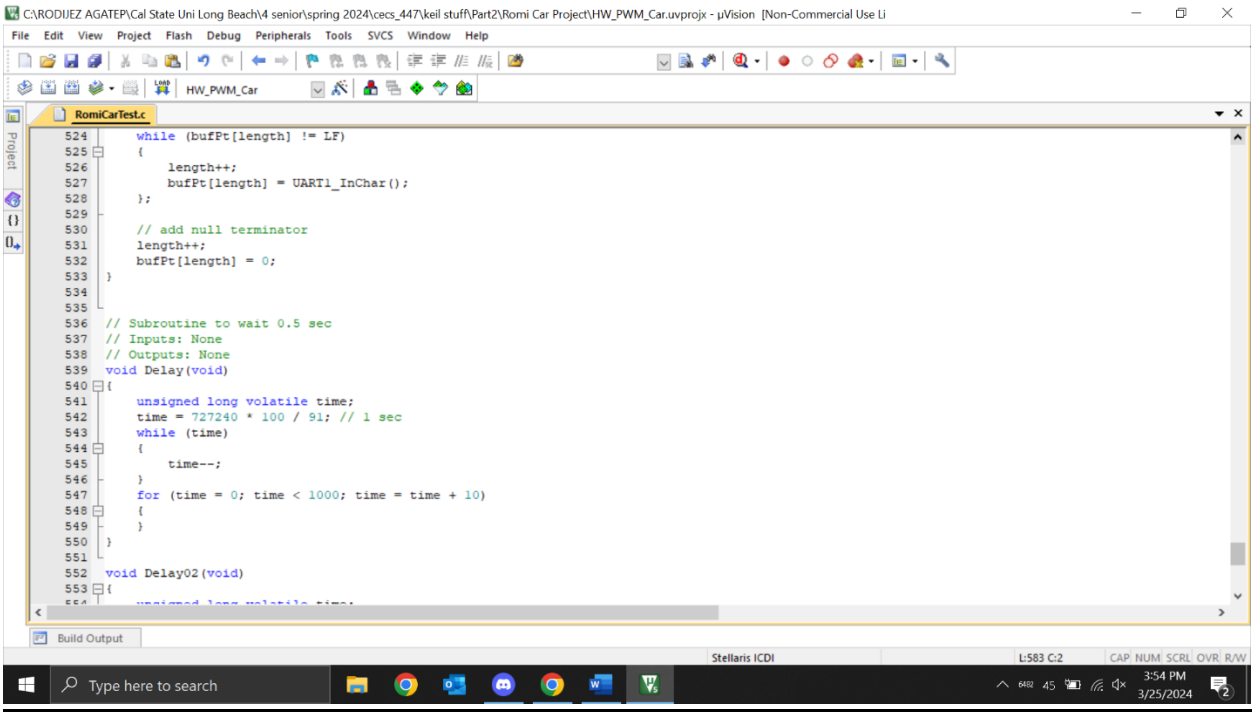

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
374 PWM0B_Duty(STOP);
375 PWM0A_Duty(current_speed);
376 PWM0B_Duty(current_speed);
377 }
378 else if(current_speed==SPEED_50){
379 current_speed=SPEED_40;
380 PWM0A_Duty(STOP);
381 PWM0B_Duty(STOP);
382 PWM0A_Duty(current_speed);
383 PWM0B_Duty(current_speed);
384 }
385 else if(current_speed==SPEED_40){
386 current_speed=SPEED_30;
387 PWM0A_Duty(STOP);
388 PWM0B_Duty(STOP);
389 PWM0A_Duty(current_speed);
390 PWM0B_Duty(current_speed);
391 }
392 break;
393
394 default:
395 break;
396 }
397 }
398 }
399 }
400
401 void GPIOPortF_Handler(void)
402 {
403 // simple debouncing code: generate 20ms to 30ms delay
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:55 PM
3/25/2024

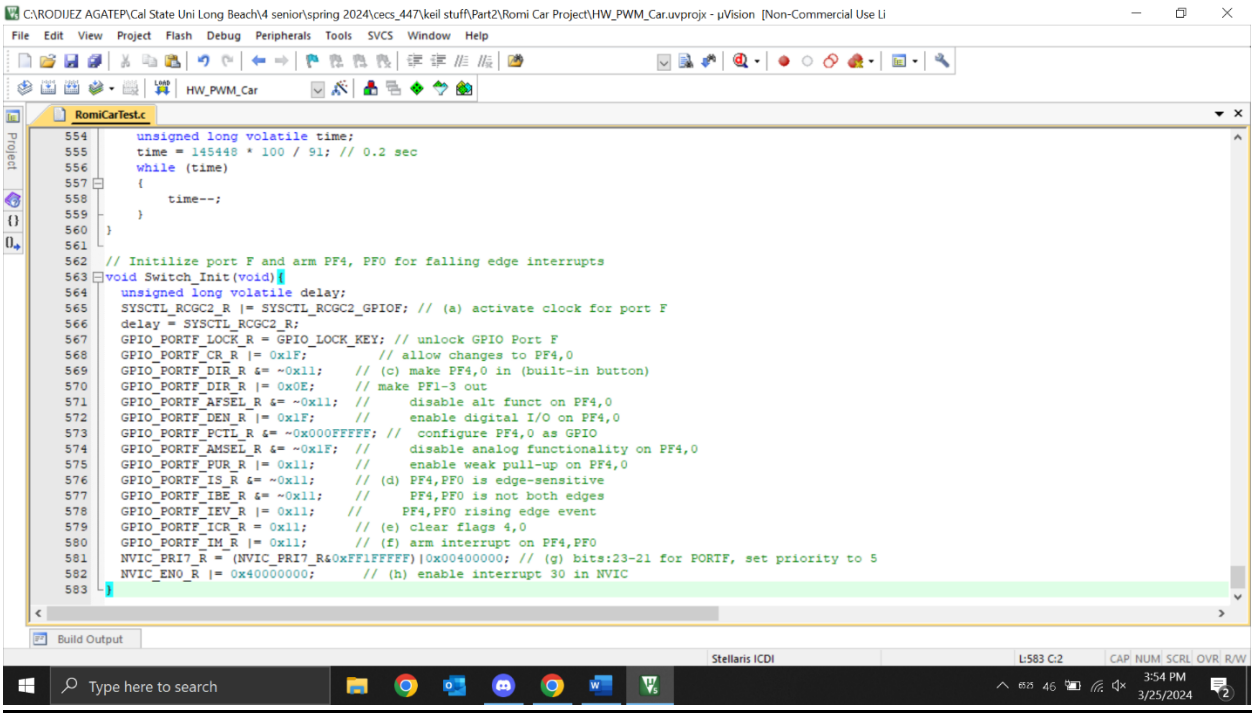

C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
404 for (uint32_t time=0;time<80000;time++) {}
405
406 if(GPIO_PORTF_RIS_R & SW2) //When SW2 is pressed on Microcontroller 1 or 2, the led will become red.
407 {
408 GPIO_PORTF_ICR_R = SW2;
409 LED=Blue;
410 model=false;
411 }
412
413 if(GPIO_PORTF_RIS_R & SW1)//If I pressed SW2 on microcontroller 1 and not 2, I can press sw1 to make microcontroller 2 red and vice versa.
414 {
415 GPIO_PORTF_ICR_R = SW1;
416 LED=Green;
417 model=true;
418 }
419 }
420
421
422 // Port F Initialization
423 void PortF_Init(void)
424 {
425 volatile unsigned long delay;
426 SYSCTL_RCGC2_R |= 0x00000020; // 1) F clock
427 delay = SYSCTL_RCGC2_R; // delay
428 GPIO_PORTF_LOCK_R = 0x4C4F434B; // 2) unlock PortF PF0
429 GPIO_PORTF_CR_R |= 0x0E; // allow changes to PF3-PF1
430 GPIO_PORTF_AMSEL_R &= ~0x0E; // 3) disable analog function
431 GPIO_PORTF_PCTL_R &= ~0x0000FFF0; // 4) GPIO clear bit PCTL
432 GPIO_PORTF_DIR_R |= 0x0E; // 6) PF1-PF3 output
433 GPIO_PORTF_AFSEL_R &= ~0x0E; // 7) no alternate function
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:55 PM
3/25/2024


C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
434 GPIO_PORTF_DEN_R |= 0x0E; // 8) enable digital pins PF3-PF1
435
436 }
437
438 //------------UART_Init------------
439 // Initialize the UART for 19200 baud rate (assuming 16 MHz UART clock),
440 // 8 bit word length, no parity bits, one stop bit, FIFOs enabled
441 // Input: none
442 // Output: none
443 void UART_Init(void)
444 {
445 // Activate Clocks
446 SYSCTL_RCGC1_R |= SYSCTL_RCGC1_UART1; // activate UART1
447 SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOB; // activate port B
448 SYSCTL_RCGC1_R |= SYSCTL_RCGC1_UART0; // activate UART0
449 SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOA; // activate port A
450
451 UART0_CTL_R &= ~UART_CTL_UARTEN; // disable UART
452 UART0_IBRD_R = 17; // IBRD = int(16,000,000 / (16 * 57600)) = int(17.3611111)
453 UART0_FBRD_R = 23; // FBRD = round(3611111 * 64) = 27
454 // 8 bit word length (no parity bits, one stop bit, FIFOs)
455 UART0_LCRH_R = (UART_LCRH_WLEN_8 | UART_LCRH_FEN);
456 UART0_CTL_R |= 0x301; // enable UART for both Rx and Tx
457
458 GPIO_PORTA_AFSEL_R |= 0x03; // enable alt funct on PA1,PA0
459 GPIO_PORTA_DEN_R |= 0x03; // enable digital I/O on PA1,PA0
460 // configure PA1,PA0 as UART0
461 GPIO_PORTA_PCTL_R = (GPIO_PORTA_PCTL_R & 0xFFFFFF00) + 0x00000011;
462 GPIO_PORTA_AMSEL_R &= ~0x03; // disable analog functionality on PA1,PA0
463
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:55 PM
3/25/2024


C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
464 UART1_CTL_R &= ~UART_CTL_UARTEN; // disable UART
465
466 // Data Communication Mode, Buad Rate = 57600
467 UART1_IBRD_R = 17; // IBRD = int(16,000,000 / (16 * 57600)) = int(17.3611111)
468 UART1_FBRD_R = 23; // FBRD = round(3611111 * 64) = 27
469
470 // 8 bit word length (no parity bits, one stop bit, FIFOs
471 // 8 bit word length (no parity bits, one stop bit, FIFOs)
472 UART1_LCRH_R = (UART_LCRH_WLEN_8 | UART_LCRH_FEN);
473 UART1_CTL_R |= 0x301; // enable UART for both Rx and Tx
474
475 GPIO_PORTB_AFSEL_R |= 0x03; // enable alt funct on PB1,PB0
476 GPIO_PORTB_DEN_R |= 0x03; // enable digital I/O on PB1,PB0
477 // configure PB1,PB0 as UART1
478 GPIO_PORTB_PCTL_R = (GPIO_PORTB_PCTL_R & 0xFFFFFF00) + 0x00000011;
479 GPIO_PORTB_AMSEL_R &= ~0x03; // disable analog functionality on PB1,PB0
480 }
481
482 //------------UART0_OutChar------------
483 // Output 8-bit to serial port
484 // Input: letter is an 8-bit ASCII character to be transferred
485 // Output: none
486 void UART0_OutChar(unsigned char data)
487 {
488 while ((UART0_FR_R & UART_FR_TXFF) != 0)
489 ;
490 UART0_DR_R = data;
491 }
492
493 //------------UART0_OutString------------
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:55 PM
3/25/2024


C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
494 // Output String (NULL termination)
495 // Input: pointer to a NULL-terminated string to be transferred
496 // Output: none
497 void UART0_OutString(unsigned char *pt)
498 {
499 while (*pt)
500 {
501 UART0_OutChar(*pt);
502 pt++;
503 }
504 }
505
506 //------------UART1_InChar------------
507 // Wait for new serial port input
508 // Input: none
509 // Output: ASCII code for key typed
510 unsigned char UART1_InChar(void)
511 {
512 while ((UART1_FR_R & UART_FR_RXFE) != 0)
513 ;
514 return ((unsigned char)(UART1_DR_R & 0xFF));
515 }
516
517 // This function reads response from HC-05 Bluetooth module.
518 void BLT_InString(unsigned char *bufPt)
519 {
520 unsigned char length = 0;
521 bufPt[length] = UART1_InChar();
522
523 // Two possible endings for a reply from HC-05: OK\r\n, ERROR:(0)\r\n
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:55 PM
3/25/2024


C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
524 while (bufPt[length] != LF)
525 {
526 length++;
527 bufPt[length] = UART1_InChar();
528 };
529
530 // add null terminator
531 length++;
532 bufPt[length] = 0;
533 }
534
535
536 // Subroutine to wait 0.5 sec
537 // Inputs: None
538 // Outputs: None
539 void Delay(void)
540 {
541 unsigned long volatile time;
542 time = 727240 * 100 / 91; // 1 sec
543 while (time)
544 {
545 time--;
546 }
547 for (time = 0; time < 1000; time = time + 10)
548 {
549 }
550 }
551
552 void Delay02(void)
553 {
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:54 PM
3/25/2024


C:\RODIJEZ AGATEP\Cal State Uni Long Beach\4 senior\spring 2024\cecs_447\keil stuff\Part2\Romi Car Project\HW_PWM_Car.uvprojx - µVision [Non-Commercial Use Li
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HW_PWM_Car
RomiCarTest.c
554 unsigned long volatile time;
555 time = 145448 * 100 / 91; // 0.2 sec
556 while (time)
557 {
558 time--;
559 }
560 }
561
562 // Initilize port F and arm PF4, PF0 for falling edge interrupts
563 void Switch_Init(void){
564 unsigned long volatile delay;
565 SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOF; // (a) activate clock for port F
566 delay = SYSCTL_RCGC2_R;
567 GPIO_PORTF_LOCK_R = GPIO_LOCK_KEY; // unlock GPIO Port F
568 GPIO_PORTF_CR_R |= 0x1F; // allow changes to PF4,0
569 GPIO_PORTF_DIR_R &= ~0x11; // (c) make PF4,0 in (built-in button)
570 GPIO_PORTF_DIR_R |= 0x0E; // make PF1-3 out
571 GPIO_PORTF_AFSEL_R &= ~0x11; // disable alt funct on PF4,0
572 GPIO_PORTF_DEN_R |= 0x1F; // enable digital I/O on PF4,0
573 GPIO_PORTF_PCTL_R &= ~0x000FFFFF; // configure PF4,0 as GPIO
574 GPIO_PORTF_AMSEL_R &= ~0x1F; // disable analog functionality on PF4,0
575 GPIO_PORTF_PUR_R |= 0x11; // enable weak pull-up on PF4,0
576 GPIO_PORTF_IS_R &= ~0x11; // (d) PF4,PF0 is edge-sensitive
577 GPIO_PORTF_IBE_R &= ~0x11; // PF4,PF0 is not both edges
578 GPIO_PORTF_IEV_R |= 0x11; // PF4,PF0 rising edge event
579 GPIO_PORTF_ICR_R = 0x11; // (e) clear flags 4,0
580 GPIO_PORTF_IM_R |= 0x11; // (f) arm interrupt on PF4,PF0
581 NVIC_PRI7_R = (NVIC_PRI7_R&0xFF1FFFFF)|0x00400000; // (g) bits:23-21 for PORTF, set priority to 5
582 NVIC_EN0_R |= 0x40000000; // (h) enable interrupt 30 in NVIC
583 }
Build Output
Stellaris ICDI
L:583 C:2
CAP NUM SCRL OVR R/W
Type here to search
3:54 PM
3/25/2024

CODE EXPLANATION

PART 1:

Main function: Initializes UART0 (serial communication) and BLT, and then enters a loop to receive and process commands for configuring the HC-05 module.

UART0 initialization: Initializes UART0 for communication with the serial terminal.

OutString: Sends strings to the serial terminal to display messages and instructions.

BLT initialization: Initializes the Bluetooth module (HC-05) for communication with the microcontroller.

Loop: Receives a command from the serial terminal using UART0_InString. Appends "\r\n" (carriage return, newline) to the command string using sprintf to format the command for the HC-05 module. Sends the formatted command to the Bluetooth module using BLT_OutString. Waits for the Bluetooth module to respond by checking the UART1 status register (UART1_FR_R) for the BUSY flag to become 0. Reads the response from the Bluetooth module using BLT_InString. Sends the response back to the serial terminal using UART0_OutString. Resets the command and response strings for the next iteration.

PART 2:

Global Defines: Defines constants for LED colors, PWM duty cycles, and other parameters.

Function Prototypes: Prototypes for functions used in the program.

Main function: Initializes PWM for the motors and LED control, initializes ports for input and output, initializes UART for Bluetooth communication, and enters a loop to receive and process commands.

UART_Init: Initializes UART for communication with the Bluetooth module.

UART0_OutChar: Sends a single character over UART0.

UART0_OutString: Sends a string over UART0.

UART1_InChar: Waits for and returns a single character received over UART1.

BLT_InString: Reads a response from the Bluetooth module until a newline character is received.

PortF_Init: Initializes Port F for onboard LEDs and switches.

Switch_Init: Initializes the switches on Port F for input.

GPIOPortF_Handler: Interrupt handler for Port F, used to change operating modes based on switch input.

PWM0A_Duty and PWM0B_Duty: Functions to set the duty cycle of PWM channels A and B.

Delay and Delay02: Functions to create delays for timing purposes.

Conclusion

The first thing we did was set up the HC05 Bluetooth module. We used example project BlueToothSetup and set it up with our PC serial terminal. Next, we tested example project BlueTooth_Controlled_LED. We then checked out the ROMI Car and tested the example code;

our initial car was not working properly, so we checked out another car. We used the previous projects in 346 and 347 for writing the Romi car code in part 2.

We demoed mode 1 and mode 2 successfully. We then demoed part 1, we lost 1 part because the query kept on repeating. The most difficult part of this project was the timing to make the car stop when making each shape.

This lab was significantly easier than project 2. It also helped that we did the Car projects during CECS 346 and 347. Those projects taught us about PWM and DC motors.