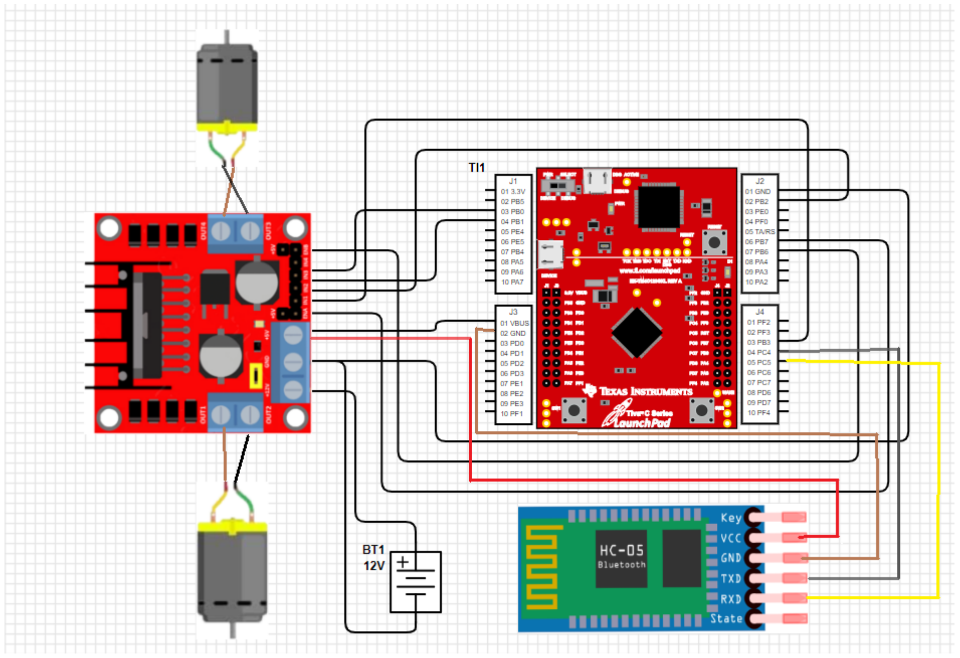
INTRODUCTION: Set up a Bluetooth module to communicate with the micro-controller using UART. Give commands through a Bluetooth module to the TM4C to drive the robot car. The system will support seven different commands.

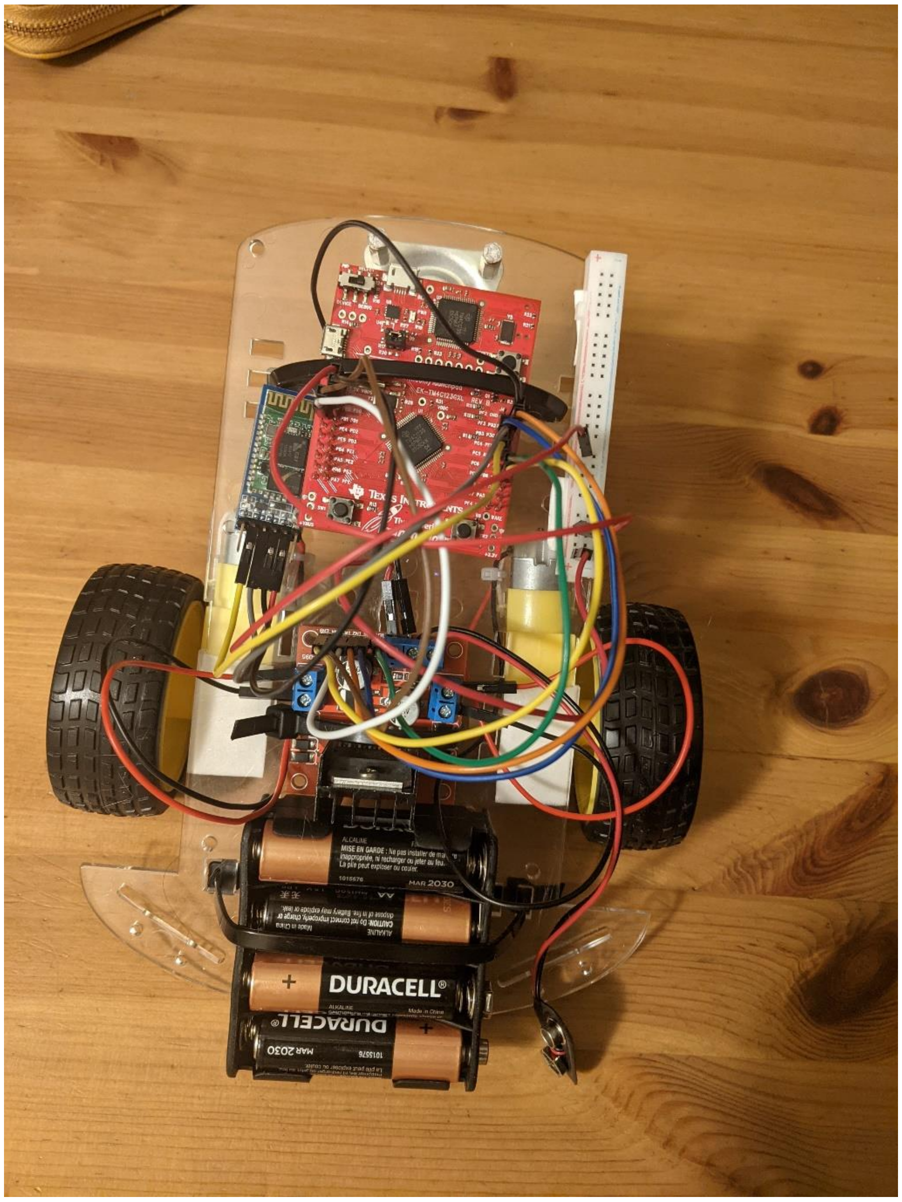
OPERATION: The operations will take place on a Bluetooth terminal, either through the PC or cell phone. The microcontroller will receive the commands and adjust the duty cycle to change the speed of the two motors. The commands include forward, reverse, left turn, right turn, stop, speed up and slow down. Each command has a corresponding color light which will be shown on the micro-controller itself, excluding the speed-up and slow-down which will adopt the previous light color. The Bluetooth setup itself will have a custom name for the Bluetooth signal, baud rate, and password. The robot car was built from a previous project as well as the foundation for a robot car execution. It includes the PLL of the system clock which will be set to generate a 50MHz system clock on the TM4C123G micro-controller. It should be noted that port C was used for the Bluetooth module as it also supported UART1.

YOUTUBE: https://www.youtube.com/watch?v=ZrdDPIrbFUQ&ab_channel=b

THEORY: The theory is an extension of UART which includes how Bluetooth operates. From this project alone it opens a lot of opportunities for projects that include communication between devices. Understanding how Bluetooth works not only was it important for the completion of this project but in the real world as well. Currently Bluetooth I imagine is one of the most used systems for communication between devices, every phone and most computers support Bluetooth capabilities. What this project revealed was that by setting up the Bluetooth module correctly for the micro-controller there was little setup between the communication between the phone and computer. This allowed great flexibility in choosing which device communicates with the micro-controller and I imagine with a little more work, there could be communication between two micro-controllers using Bluetooth with little effort.

HARDWARE DESIGN:





SOFTWARE DESIGN:

BLUETOOTH SETUP CODE:

```
// main.c

// Runs on LM4F120/TM4C123

// this connection occurs in the USB debugging cable

// U1Rx (PB0)

// U1Tx (PB1)
// Ground connected ground in the USB cable
// Header files
#include "tm4c123gh6pm.h"
#include "UART.h"
#include "string.h"
#define RED                 0x02
#define GREEN               0x08
#define BLUE                0x04
#define PURPLE              0x06
#define WHITE  0x0E
#define DARK                0x00
void Delay(void);
void PortF_Init(void);
char Color;
// main function for programming BT device with no UI
//int mainNoUI(void) {
int main(void) {
        UART_Init();
        PortF_Init();
//char String[30];
//  // setup the HC-05 bluetooth module
//                  UART0_OutString("AT+NAME=Tylor Blue\r\n");             // Name = Tylor Blue
//                  UART1_OutString("AT+NAME=Tylor Blue\r\n");             // Name = Tylor Blue
//    while ((UART1_FR_R&UART_FR_BUSY) != 0){};
//    BLT_InString(String);
//    UART0_OutString(String);
//
//                  Delay();
//
```

```
//			UART0_OutString("AT+UART=57600,0,1,\r\n"); // baud rate = 57600, 1 stop bit, odd
parity
//			UART1_OutString("AT+UART=57600,0,1,\r\n"); // baud rate = 57600, 1 stop bit, odd
parity
//	while ((UART1_FR_R&UART_FR_BUSY) != 0){};
//	BLT_InString(String);
//	UART0_OutString(String);
//
//			Delay();
//
//			UART0_OutString("AT+PSWD=1122\r\n");			// Pass = 0824
//			UART1_OutString("AT+PSWD=1122\r\n");			// Pass = 0824
//	while ((UART1_FR_R&UART_FR_BUSY) != 0){};
//	BLT_InString(String);
//	UART0_OutString(String);
//
//			Delay();
//
//			UART0_OutString("AT+ROLE=0\r\n");				// Mode = Slave
//			UART1_OutString("AT+ROLE=0\r\n");				// Mode = Slave
//	while ((UART1_FR_R&UART_FR_BUSY) != 0){};
//	BLT_InString(String);
//	UART0_OutString(String);


//			Delay();
//
//  // query the HC-05 bluetooth module
//			UART0_OutString("AT+UART?\r\n"); // baud rate = 57600, 1 stop bit, odd parity
//			UART1_OutString("AT+UART?\r\n"); // baud rate = 57600, 1 stop bit, odd parity
```

```
//  while ((UART1_FR_R&UART_FR_BUSY) != 0){};
//  BLT_InString(String);
//  UART0_OutString(String);
//
//              Delay();
//
//              UART0_OutString("AT+PSWD?\r\n");            // Pass = 0824
//              UART1_OutString("AT+PSWD?\r\n");            // Pass = 0824
//  while ((UART1_FR_R&UART_FR_BUSY) != 0){};
//  BLT_InString(String);
//  UART0_OutString(String);
//
//              Delay();
//
//              UART0_OutString("AT+ROLE?\r\n");                            // Mode = Slave
//              UART1_OutString("AT+ROLE?\r\n");                            // Mode = Slave
//  while ((UART1_FR_R&UART_FR_BUSY) != 0){};
//  BLT_InString(String);

//              Delay();
//
//  UART0_OutString(String);
//              UART0_OutString("AT+NAME?\r\n");           // Name = CECS447
//              UART1_OutString("AT+NAME?\r\n");           // Name = CECS447
//  while ((UART1_FR_R&UART_FR_BUSY) != 0){};
//  BLT_InString(String);
//  UART0_OutString(String);
//
//              UART0_OutString(String);
```

```
//			UART0_OutString("AT+VERSION?\r\n");		// Name = CECS447
//			UART1_OutString("AT+VERSION?\r\n");		// Name = CECS447
//	while ((UART1_FR_R&UART_FR_BUSY) != 0){};
//	BLT_InString(String);
//	UART0_OutString(String);

  while (1) {

			Color = UART1_InChar();
			switch (Color){
				case 'r' : GPIO_PORTF_DATA_R = RED;
					break;
				case 'b' : GPIO_PORTF_DATA_R = BLUE;
					break;
				case 'g' : GPIO_PORTF_DATA_R = GREEN;
					 break;
				case 'p' : GPIO_PORTF_DATA_R = PURPLE;
					break;
				case 'w' : GPIO_PORTF_DATA_R = WHITE;
					break;
				case 'd' : GPIO_PORTF_DATA_R = DARK;
					break;
				default : GPIO_PORTF_DATA_R = DARK;
			}
}
}

void PortF_Init(){
		SYSCTL_RCGC2_R |= 0x20;
```

```
            // Port F_Buttons


  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock PortF PF0
            GPIO_PORTF_CR_R |= 0x1F;                              // allow changes to PF4-0 :11111->0x1F
  GPIO_PORTF_AMSEL_R &= ~0x1F;        // disable analog function
  GPIO_PORTF_PCTL_R &= ~0x000FFFFF;             // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R &= ~0x11;          // PF4,PF0 input
  GPIO_PORTF_DIR_R |= 0x0E;                 // PF3,PF2,PF1 output
            GPIO_PORTF_AFSEL_R &= ~0x1F;        // no alternate function
  GPIO_PORTF_PUR_R |= 0x11;               // enable pullup resistors on PF4,PF0
  GPIO_PORTF_DEN_R |= 0x1F;               // enable digital pins PF4-PF0


            GPIO_PORTF_IS_R                             &= ~0x11;       // PF4,0 is edge-sensitive
            GPIO_PORTF_IBE_R                            &= ~0x11;       // PF4,0 is not both edges
            GPIO_PORTF_IEV_R                            &= ~0x11;                       // PF4,0 falling edge event
            GPIO_PORTF_ICR_R                            |= 0x11;             // clear flag4,0
            GPIO_PORTF_IM_R                             |= 0x11;             // arm interrupt on PF4,0
            NVIC_PRI7_R                                             |= (NVIC_PRI7_R&0xFF1FFFFF)|0x0020000;
//priority 1
            NVIC_EN0_R                                                          |= 0x40000000;      // enable interrupt
30 in NVIC
}
void Delay(){
            unsigned long volatile time;
  time = 2000000;  // 0.1sec
  while(time){
                        time--;
  }
}
```

ROBOT CAR CODE:

```
// Documentation
// CECS447 project 4 3: Bluetooth Controlled Robot Car
// Description: Use bluetooth module and uart to communicate with robot car
// Student Name: Tylor Cooks 026538081
// YOUTUBE VIDEO: https://www.youtube.com/watch?v=ZrdDPIrbFUQ&ab_channel=b
#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "UART.h"
#include "string.h"
#define SYSCTL_RIS_R            (*((volatile unsigned long *)0x400FE050))
#define SYSCTL_RIS_PLLLRIS      0x00000040  // PLL Lock Raw Interrupt Status
#define SYSCTL_RCC_R            (*((volatile unsigned long *)0x400FE060))
#define SYSCTL_RCC_XTAL_M       0x000007C0  // Crystal Value
#define SYSCTL_RCC_XTAL_6MHZ    0x000002C0  // 6 MHz Crystal
#define SYSCTL_RCC_XTAL_8MHZ    0x00000380  // 8 MHz Crystal
#define SYSCTL_RCC_XTAL_16MHZ   0x00000540  // 16 MHz Crystal
#define SYSCTL_RCC2_R           (*((volatile unsigned long *)0x400FE070))
#define SYSCTL_RCC2_USERCC2     0x80000000  // Use RCC2
#define SYSCTL_RCC2_DIV400      0x40000000  // Divide PLL as 400 MHz vs. 200
                                            // MHz
#define SYSCTL_RCC2_SYSDIV2_M   0x1F800000  // System Clock Divisor 2
#define SYSCTL_RCC2_SYSDIV2LSB  0x00400000  // Additional LSB for SYSDIV2
#define SYSCTL_RCC2_PWRDN2      0x00002000  // Power-Down PLL 2
#define SYSCTL_RCC2_BYPASS2     0x00000800  // PLL Bypass 2
#define SYSCTL_RCC2_OSCSRC2_M   0x00000070  // Oscillator Source 2
#define SYSCTL_RCC2_OSCSRC2_MO  0x00000000  // MOSC
#define SYSDIV2 7 // PLL 50MHz
#define DIRECTION       (*((volatile unsigned long *)0x400053FC)) //PORTB DATA REGISTER
```

```
#define LIGHT                                        (*((volatile unsigned long *)0x400253FC)) //PORTF DATA
REGISTER

#define PERIOD                   25000

#define F_B 0x0A //forward backward


#define RED                 0x02

#define GREEN               0x08

#define YELLOW  0x0A

#define BLUE                0x04

#define PURPLE              0x06

#define WHITE  0x0E

#define DARK                0x00

extern void DisableInterrupts(void); //Disable interrupts

extern void EnableInterrupts(void);  //Enable interrupts

extern void WaitForInterrupt(void);  //low power mode


void PLL_Init(void);

void PWM0_0_Init(unsigned long dutyL, unsigned long dutyR);

void PortF_Init(void);

void PortB_Init(void);

void debounce(void);

unsigned long H = PERIOD/2;

unsigned long speed_count;

unsigned long dFlag; // direction flag

const unsigned long Delta = 2500;

unsigned short Speed = 0;

unsigned char Input;

int main(void){
```

```
PLL_Init();

UART_Init();

PortF_Init();

PortB_Init();


while(1){

Input= UART1_InChar();// UART1_OutChar(Input);

        switch(Input){

                case 'w':

                        GPIO_PORTF_DATA_R = GREEN;

                        GPIO_PORTB_DATA_R &= ~0x0F;

                        GPIO_PORTB_DATA_R |= F_B;

                        PWM0_0_Init(H/2,H/2);

                        break;

                case 's':

                        GPIO_PORTF_DATA_R = BLUE;

                        GPIO_PORTB_DATA_R &= ~0x0F;

                        GPIO_PORTB_DATA_R |= ~F_B;

                        PWM0_0_Init(H/2,H/2);

                        break;

                case 'a':

                        GPIO_PORTF_DATA_R = YELLOW;

                        PWM0_0_Init(H/3,H/4);

                        break;

                case 'd':

                        GPIO_PORTF_DATA_R = PURPLE;

                        PWM0_0_Init(H/4,H/3);

                        break;

                case 't':
                        GPIO_PORTF_DATA_R = DARK;
                        PWM0_0_Init(0,0);
                        break;
                case 'l':
                        if(H/2>Delta) H = 2*(H/2 - Delta);
                        PWM0_0_Init(H/2,H/2);
                        break;
                case 'u':
                        if(H/2<(PERIOD-Delta)) H = 2*(H/2 + Delta);
                        PWM0_0_Init(H/2,H/2);
                        break;
            }

        }

}
void PortF_Init(void){
        volatile unsigned long delay;
        SYSCTL_RCGC2_R |= 0x00000020;
        delay = SYSCTL_RCGC2_R;
        GPIO_PORTF_LOCK_R = 0x4C4F434B; // unlock portf for port 0
        GPIO_PORTF_CR_R |= 0x1F;
        GPIO_PORTF_AMSEL_R &= ~0x1E; // disable analog for PF4-0
        GPIO_PORTF_PCTL_R &= ~0x000FFFFF; // GPIO clear PCTL bits for PF4-0
        GPIO_PORTF_DIR_R |= 0x0E;
        GPIO_PORTF_DIR_R &= ~0x11; //
        GPIO_PORTF_AFSEL_R &= ~0x1F;
        GPIO_PORTF_PUR_R |= 0x11;
        GPIO_PORTF_DEN_R |= 0x1F;
```

```
	//interrupt stuff
	GPIO_PORTF_IS_R &= ~0x11;     // (d) PF4 is edge-sensitive
	GPIO_PORTF_IBE_R &= ~0x11;    //     PF4 negative edge detect
	GPIO_PORTF_IEV_R &= ~0x11;    //
	GPIO_PORTF_ICR_R = 0x11;      // (e) clear flag4
	GPIO_PORTF_IM_R |= 0x11;      // (f) arm interrupt on PF1 AND PF4
	NVIC_PRI7_R = (NVIC_PRI7_R&0xFF1FFFFF)|0x00400000; // (g) priority 2
	NVIC_EN0_R |= 0x40000000;      // (h) enable interrupt 30 in NVIC

}
void PortB_Init(void){
	unsigned long volatile delay;
	SYSCTL_RCGC2_R |= 0x00000002;
	GPIO_PORTB_AMSEL_R &= ~0x0F; //disable analog functionality on PB0-3
	GPIO_PORTB_PCTL_R  &= ~0x0000FFFF;
	GPIO_PORTB_DIR_R   |= 0x0F; // 1 = ouput , PB0-3
	GPIO_PORTB_AFSEL_R &= ~0x0F; //disable alt funct on PB0-3
	GPIO_PORTB_DEN_R   |= 0x0F; // enable data pins
}

void debounce(void){
	unsigned long i;
	for(i = 0; i < 60000; ++i);


}
void PWM0_0_Init(unsigned long dutyL, unsigned long dutyR){//PB7-6
	SYSCTL_RCGCPWM_R |= 0X00000001; //enable PWM module 0 clock source
	while((SYSCTL_PRGPIO_R&0x20)==0);
```

```
        GPIO_PORTB_AFSEL_R |= 0xC0; //enable PB7-6 as alternate function output
        GPIO_PORTB_DEN_R |= 0xC0;
        GPIO_PORTB_PCTL_R |= (GPIO_PORTB_PCTL_R&(~0xFF000000))|0x44000000;//enable
M0PWM1-0 on pins PB7-6
        SYSCTL_RCC_R  = (SYSCTL_RCC_R&(~0x000E0000))|0x00100000; //clear bits 17-19 of PWMDIV
for /2 divisor and enable USEPWMDIV (bit 20)
        PWM0_0_CTL_R  = 0; //count-down mode
        PWM0_0_GENB_R = 0xC08; //low on LOAD, high on CMPB down
        PWM0_0_GENA_R = 0xC8;  //low on LOAD, high on CMPA down
        PWM0_0_LOAD_R = PERIOD - 1;          // 5) cycles needed to count down to 0
  PWM0_0_CMPB_R = dutyL - 1;            // 6) count value when output rises
        PWM0_0_CMPA_R = dutyR        - 1;            // 6) count value when output rises
        PWM0_0_CTL_R |= 0x00000001; //enable PWM module 0
        PWM0_ENABLE_R |= 0x00000003;         //enable PWM0/M0PWM1-0
}
void PLL_Init(void){
  SYSCTL_RCC2_R |= SYSCTL_RCC2_USERCC2;   //Step 0:  Allow usage of RCC2 for advanced clocking
features
        SYSCTL_RCC2_R |= SYSCTL_RCC2_BYPASS2;   //Step 1:  BYPASS PLL... clock divider not in use
        SYSCTL_RCC_R  &= ~SYSCTL_RCC_XTAL_M;    //Step 2:  clear XTAL field
  SYSCTL_RCC_R  += SYSCTL_RCC_XTAL_16MHZ; //        configure XTAL for 16 MHz crystal
  SYSCTL_RCC2_R &= ~SYSCTL_RCC2_OSCSRC2_M;//        clear oscillator source field
  SYSCTL_RCC2_R += SYSCTL_RCC2_OSCSRC2_MO;//        configure for main oscillator source
        SYSCTL_RCC2_R &= ~SYSCTL_RCC2_PWRDN2;   //Step 3:  clear PWRDN2 bit to activate PLL
        SYSCTL_RCC2_R |= SYSCTL_RCC2_DIV400;    //        use 400 MHz PLL
  SYSCTL_RCC2_R = (SYSCTL_RCC2_R&~0x1FC00000) //Step 4: clear system clock divider field
          + (SYSDIV2<<22);       //           configure for 50 MHz clock
        while((SYSCTL_RIS_R&SYSCTL_RIS_PLLLRIS)==0);//Step 5: wait for the PLL to lock by polling
PLLLRIS
        SYSCTL_RCC2_R &= ~SYSCTL_RCC2_BYPASS2;  //Step 6: Clear BYPASS
```

```
}
```

UART CODE:

```
// 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
#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "UART.h"
//------------UART_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 UART_Init(void){
        // Activate Clocks
  SYSCTL_RCGC1_R |= SYSCTL_RCGC1_UART1; // activate UART1
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOC; // activate port B
        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

  UART1_CTL_R &= ~UART_CTL_UARTEN;      // disable UART

            // Command Mode, default Buad Rate for HC-05 command mode = 38400
  UART1_IBRD_R = 54;                  // IBRD = int(16,000,000 / (16 * 38400)) = int(26.04166667)
  UART1_FBRD_R = 16;                   // 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_PORTC_AFSEL_R |= 0x30;          // enable alt funct on PB1,PB0
  GPIO_PORTC_DEN_R |= 0x30;            // enable digital I/O on PB1,PB0
                                // configure PB1,PB0 as UART1
  GPIO_PORTC_PCTL_R = (GPIO_PORTC_PCTL_R&0xFF00FFFF)+0x00220000;
  GPIO_PORTC_AMSEL_R &= ~0x30;         // disable analog functionality on PB1,PB0
}

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

//------------UART1_OutChar------------
// Output 8-bit to serial port
// Input: letter is an 8-bit ASCII character to be transferred
// Output: none
void UART1_OutChar(unsigned char data){
  while((UART1_FR_R&UART_FR_TXFF) != 0);
  UART1_DR_R = data;
}

//------------UART0_OutString------------
// Output String (NULL termination)
// Input: pointer to a NULL-terminated string to be transferred
// Output: none
void UART0_OutString(char *pt){
  while(*pt){
    UART0_OutChar(*pt);
    pt++;
  }
}

//------------UART1_OutString------------
// Output String (NULL termination)
// Input: pointer to a NULL-terminated string to be transferred
// Output: none
void UART1_OutString(char *pt){
  while(*pt){
    UART1_OutChar(*pt);
    pt++;
  }
}



//------------UART0_InChar------------
// Wait for new serial port input
// Input: none
// Output: ASCII code for key typed
unsigned char UART0_InChar(void){
  while((UART0_FR_R&UART_FR_RXFE) != 0);
  return((unsigned char)(UART0_DR_R&0xFF));
}



//------------UART1_InChar------------
// Wait for new serial port input
// Input: none
// Output: ASCII code for key typed
unsigned char UART1_InChar(void){
  while((UART1_FR_R&UART_FR_RXFE) != 0);
  return((unsigned char)(UART1_DR_R&0xFF));
}

//------------UART0_InString------------
// Accepts ASCII characters from the serial port
//    and adds them to a string until <enter> is typed
//    or until max length of the string is reached.
// It echoes each character as it is inputted.
// If a backspace is inputted, the string is modified
//    and the backspace is echoed
// terminates the string with a null character
// uses busy-waiting synchronization on RDRF
// Input: pointer to empty buffer, size of buffer
// Output: Null terminated string
// -- Modified by Agustinus Darmawan + Mingjie Qiu --
int UART0_InString(char *bufPt, uint16_t max) {
int length=0;
char character;
  character = UART0_InChar();
  while(character != CR){
    if(character == BS){
      if(length){
        bufPt--;
        length--;
        UART0_OutChar(BS);
      }
    }
    else if(length < max){
      *bufPt = character;
      bufPt++;
      length++;
      UART0_OutChar(character);
    }
    character = UART0_InChar();
  }
  *bufPt = 0;
```

```
        return length;
}



// This function reads response from HC-05 Bluetooth module.
void BLT_InString(char *bufPt) {
  unsigned char length=0;
  bufPt[length] = UART1_InChar();

  // Two possible ending for a reply from HC-05: OK\r\n, ERROR:(0)\r\n
  while ((bufPt[length]!='K')&&(bufPt[length]!=')')) {
    length++;
    bufPt[length] = UART1_InChar();
  };

  // take care of ending CR, LF
  length++;
  bufPt[length] = UART1_InChar();
  length++;
  bufPt[length] = UART1_InChar();

  // add null terminator
  length++;
  bufPt[length] = 0;
}
```

CONCLUSION:

Overall this project was very difficult but very informative on the workings of the Bluetooth module. The hardest part on making the Bluetooth work on the robot car was realizing that because the PLL changed the system clock to 50Mhz you had to adjust the baud rate accordingly, which was different form the initial set up. Once the Bluetooth could communicate with the TM4C it was as easy as using a switch statement to adjust the duty cycle to correspond with the commands. This project was very probably one of my favorite projects to work on.