Name: Donavan Tran and Hannah Ho
Final Project Lab Report

1. Procedure

After weeks of research, we decided to build an RC (Remote Controlled) Car as the final project for this lab. For starter, We went over the lab samples in previous course to obtain better insight on the technicality and prerequisite components to develop our own version of RC car. The project is broken down into two tasks, assigned to each individual. One works on the bluetooth, which receives input signal from an Android device, the other works on integrating the motors and drivers onto the Tiva board.

For the bluetooth, we used the MIT App Inventor 2 - an open-source app design server - and UART to build the controller for the RC car.

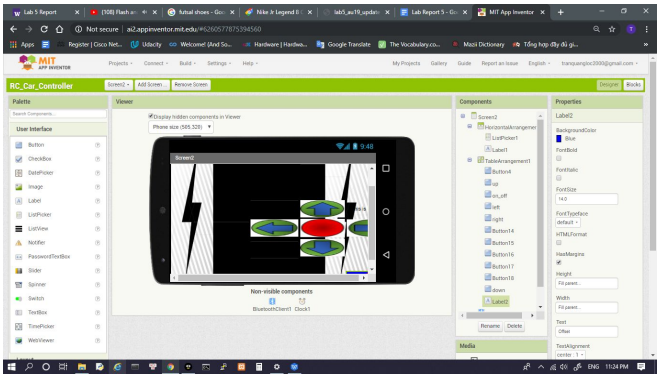


Figure 1 demonstrates how the RC car controller is designed. The top right corner of this server presented two platforms: Designers and Blocks, all of which are important to build the RC car controller. At the Designer Platform, all the backgrounds, color, fonts, texts, and buttons are added on the simulated phone screen.

Fig.1: MIT App Inventor 2 Designer Platform

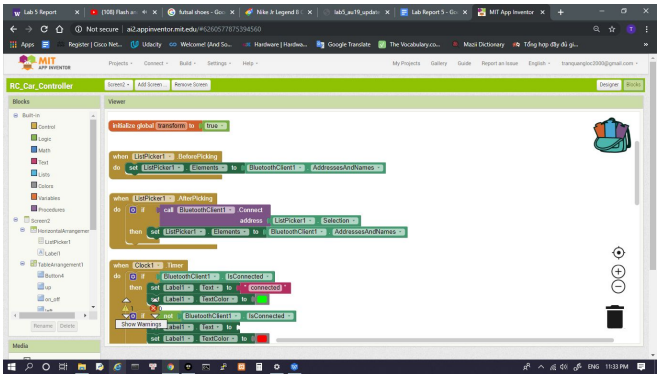


Figure 2 above describes how all the logic and bluetooth android module are implemented. Just like a puzzle, each button, text, and color added in the Designer Platforms can placed on top of each other to form proper logics when getting invoked. Whenever the controller button is pressed, it will send a corresponding command to the bluetooth module on the Tiva board.

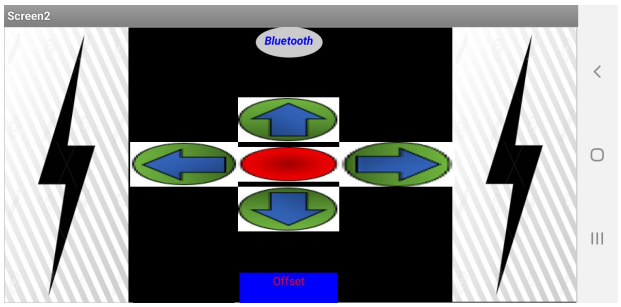
Fig.2: MIT App Inventor 2 Block Platform

For the second task, we obtained the motors and drivers on Amazon. To understand how they work, we used the product datasheet, which is available online. Next, we collectively wired them up on the Tiva Board, along with the car chassis and the bluetooth.

Finally, after all the hardware components are set up, we start debugging each written module to ensure they all work properly before assembling all together.

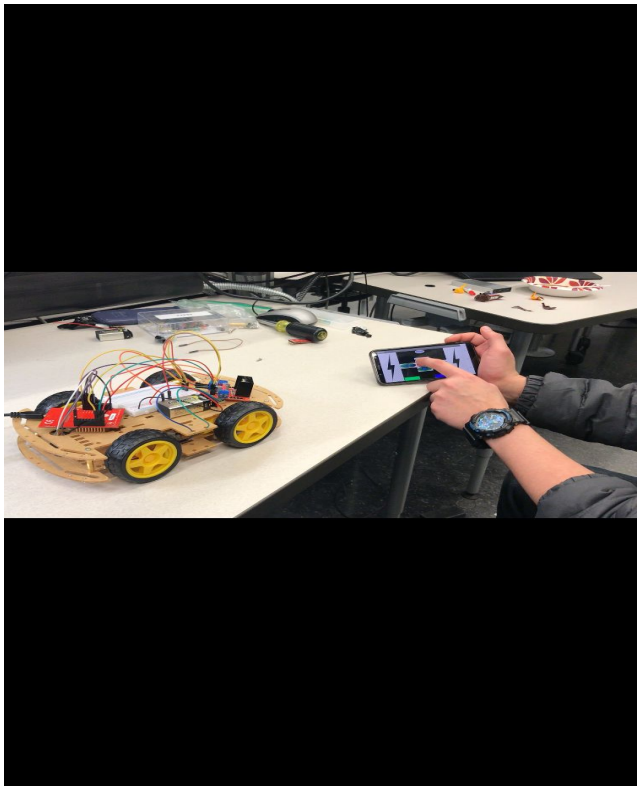
2. Results

- Bluetooth RC Controller



The background and control buttons are drawn using powerpoint. The blue box below the controller is used to keep track of which buttons are pressed during debugging.

- Driver, Motor, and Car Chassis



The whole system, which comprises the RC car, the bluetooth module, and the Android app, that overall fulfills all the motion-driven functionality of the normal car. As the bluetooth sends the signals to the bluetooth thru the android app, the signals are then sent to the driver module using Tiva board to control the motors.

3. Problems Encountered and Feedback

We encountered a lot of problems in the final project, some of which took us a tremendous amount of time to figure out before they actually work.

For the bluetooth module, it took us one full week to figure out how to implement the bluetooth on android device using MIT app inventor. Background and color design aside, the code block for the android device was a huge problem since none of us have had experience with android development before. We were able to draw the background, draft the code, and complete the controller purely using the tutorial on the website. In addition, we have to test the receiving transmission from the Android to Tiva using UART, which requires more reading from the datasheet.

For the driver module, it took us a while to figure out how to use the L298N driver at first and use GPIO pins on the Tiva Board to send signals to the driver so it could control the motors. Since the input pin of L298N takes either 5V or ground to control the motors so we used GPIO Port A2, A3 and A4 to send either 0 for Ground or 1 for positive. We used the oscilloscope to test PA2, 3 and 4 to see if the voltage output would change when we set them to either high or low.

One tip I would recommend for the final project is to start and plan early, instead of waiting until the last minute. It took us a while to obtain all the required components, not to mention the time spent on testing the product usability. Also, consult the planned project either with the instructor or the TAs to make sure all extra modules that you plan to implement are eligible and executable. In total, we spent more than 2 weeks to finish the lab.

Conclusion

Although the design of RC car contributes naught to the world's contemporary development. Yet, the modules made up of the RC car can be used in larger part of a complex system, such as a surveillance multi-terrain car, which uses a driver module to control the motors of the device and a bluetooth module to communicate wirelessly with the car by receiving and transmitting data in full-duplex mode. I believe RC car serves as a subsystem for a more complicated system.

Appendix

Main.c file

```c
#include <header_file.h>
#include <stdint.h>

char a = 'a';

// move the car in forward direction
void forward(){
  GPIODATA_PA |= 0xc;
}

// move the car in backward direction
void backward(){
  GPIODATA_PA |= 0x10;
}

// make the car turn left in forward direction by making the wheels on the right side go forward and stop the wheels on the left side
void left(){
  GPIODATA_PA |= 0x8;
  for (int j = 0 ; j < 10000000; j++) {};
  GPIODATA_PA &= ~0x8;
}

// make the car turn right in forward direction by making the wheels on the left side go forward and stop the wheels on the right side
void right(){
  GPIODATA_PA |= 0x4;
  for (int j = 0 ; j < 10000000; j++) {};
  GPIODATA_PA &= ~0x4;
}

// stop the car
void stop(){
  GPIODATA_PA = 0;
}


void GPIO_Init() {
  volatile unsigned long delay;
  RCGC2_PA |= 0x01;                     // activate clock for Port A
  delay = RCGC2_PA;                     // allow time for clock to start
  GPIOPCTL_PA &= ~0xFFF00;              // regular GPIO PA2
  GPIOAMSEL_PA &= ~0x1c;                 // disable analog  function of PA2
  GPIODIR_PA |= 0x1c;                    // set PA2 to output
  GPIOAFSEL_PA &= ~0x1c;                 // regular port function
  GPIODEN_PA |= 0x1c;                    // enable digital output on PA2

  RCGCGPIO |= 0x2;                      // enable clock to PortB
```

```
  GPIOAFSEL_PB |= 0x3;                    // allow alternative function for PB
0&1
  GPIODEN_PB |= 0x3;                      // enable digital pins
  GPIO_PCTL_PORTB |= 0x11;                // set alternative function to UART
}

void UART_Init() {
  RCGCUART |= 0x2;                        // enable clock to UART Module 1
  int32_t delay = RCGCUART;               // delay to wait for clock setup

  UART_CTL &= ~(1 << 0);                  // diable module before configuration

  // set the baud rate (integer & fraction parts)
  UART_IBRD = 8;
  UART_FBRD = 44;

  UART_LCRH |= 0x60;                      // set the transmission line control
  UART_CC = 0x0;                          // use system clock

  UART_CTL |= (1 << 9);                   // set the receive enable
  UART_CTL |= (1 << 8);                   // set the transmit enable
  UART_CTL |= (1 << 0);                   // enable the UART
}

void delay() {
  for (int i = 0; i < 1000000; i++) {}
}

// read the received data from the bluetooth using UART
char readData(void) {
    char c;
    while ((UARTFR & (1 << 4)) != 0) {}         // dont read when the receiver
is empty
    c = UARTDR;
    return c;
}

int main()
{
  GPIO_Init();
  UART_Init();

  while(1){

    delay();
    a = readData();

    // control the car using the received data from the bluetooth
    if(a == 'F'){
      forward();
```

```
      }else if (a == 'B'){
        backward();
      }else if (a == 'R'){
        right();
      }else if (a == 'L'){
        left();
      }else if (a == 'O'){
        stop();
      }
    }

    return 0;
  }
```

Header.h file

```
  #ifndef HEADER_FILE
  #define HEADER_FILE

  //LAB 1
  #define RCGCGPIO (*((volatile unsigned long*) 0x400FE608))
  #define GPIODEN (*((volatile unsigned long*) 0x4002551C))
  #define GPIODIR (*((volatile unsigned long*) 0x40025400))
  #define GPIODATA (*((volatile unsigned long*) 0x400253FC))
  #define GPIOF 0x00000020
  #define GPIOLOCK (*((volatile unsigned long*) 0x40025520))
  #define GPIOCR (*((volatile unsigned long*) 0x40025524))
  #define GPIOPUR (*((volatile unsigned long*) 0x40025510))
  #define KEY_2_UNLOCK 0x4C4F434B

  #define RCGC2_PA (*((volatile unsigned long*) 0x400FE108))
  #define GPIOPCTL_PA (*((volatile unsigned long*) 0x4000452C))
  #define GPIOAMSEL_PA (*((volatile unsigned long*) 0x40004528))
  #define GPIODIR_PA (*((volatile unsigned long*) 0x40004400))
  #define GPIOAFSEL_PA (*((volatile unsigned long*) 0x40004420))
  #define GPIODEN_PA (*((volatile unsigned long*) 0x4000451C))
  #define GPIODATA_PA (*((volatile unsigned long*) 0x400043FC))

  //LAB 2
  #define RCGCTIMER (* ((volatile unsigned long*) 0x400FE604))    // enable this
  timer register
  #define ENABLE_TIMER_0 0x00000001                               // key to
  unable the 16/32 bit general purpose timer 0
  #define GPTMCTL (* ((volatile unsigned long*) 0x4003000C))
  #define GPTMCFG (* ((volatile unsigned long*) 0x40030000))
  #define GPTMTAMR (* ((volatile unsigned long*) 0x40030004))
  #define GPTMTAILR (* ((volatile unsigned long*) 0x40030028))
  #define GPTMRIS (* ((volatile unsigned long*) 0x4003001C))
  #define GPTMICR (* ((volatile unsigned long*) 0x40030024))
```

```
#define GPTMIMR (* ((volatile unsigned long*) 0x40030018))
#define EN0 (* ((volatile unsigned long*) 0xE000E100))
#define DISn (* ((volatile unsigned long*) 0xE000E180))

#define GPIOF_Interrupt_Mask (* ((volatile unsigned long*) 0x40025410))
#define GPIOF_Interrupt_Clear (* ((volatile unsigned long*) 0x4002541C))

#define GPIO_Interrupt_Mask_PA (* ((volatile unsigned long*) 0x40004410))
#define GPIO_Interrupt_Clear_PA (* ((volatile unsigned long*) 0x4000441C))

//LAB3
#define RCGCADC (* ((volatile unsigned long*) 0x400FE638))
#define GPIOAFSEL_PE (* ((volatile unsigned long*) 0x40024420))
#define GPIOAFSEL_PD (* ((volatile unsigned long*) 0x40007420))
#define GPIOAFSEL_PB (* ((volatile unsigned long*) 0x40005420))

#define GPIODEN_PE (* ((volatile unsigned long*) 0x4002451C))
#define GPIODEN_PD (* ((volatile unsigned long*) 0x4000751C))
#define GPIODEN_PB (* ((volatile unsigned long*) 0x4000551C))

#define GPIOAMSEL_PE (* ((volatile unsigned long*) 0x40024528))
#define GPIOAMSEL_PD (* ((volatile unsigned long*) 0x40007528))
#define GPIOAMSEL_PB (* ((volatile unsigned long*) 0x40005528))

#define ADCACTSS (* ((volatile unsigned long*) 0x40038000))
#define ADCEMUX (* ((volatile unsigned long*) 0x40038014))
#define ADCSSMUX (* ((volatile unsigned long*) 0x400380A0))
#define ADCSSCTL3 (* ((volatile unsigned long*) 0x400380A4))
#define ADCACTSS (* ((volatile unsigned long*) 0x40038000))
#define ADCPSSI (* ((volatile unsigned long*) 0x40038028))
#define ADCRIS (* ((volatile unsigned long*) 0x40038004))
#define ADCSSFIFO3 (* ((volatile unsigned long*) 0x400380A8))
#define ADCISC (* ((volatile unsigned long*) 0x4003800C))
#define RCC (* ((volatile unsigned long*) 0x400FE060))
#define RIS (* ((volatile unsigned long*) 0x400FE050))
#define ADCIM (* ((volatile unsigned long*) 0x40038008))
#define RCC2 (* ((volatile unsigned long*) 0x400FE070))
#define UARTDMACTL (* ((volatile unsigned long*) 0x4000C048))
#define RCGCUART (* ((volatile unsigned long*) 0x400FE618))
#define UARTCTL (* ((volatile unsigned long*) 0x4000C030))
#define UARTIBRD (* ((volatile unsigned long*) 0x4000C024))
#define UARTFBRD (* ((volatile unsigned long*) 0x4000C028))
#define UARTFBRD (* ((volatile unsigned long*) 0x4000C028))
#define GPIOAFSEL_PA (* ((volatile unsigned long*) 0x40004420))
#define PRGPIO (* ((volatile unsigned long*) 0x400FEA08))
#define UARTLCRH (* ((volatile unsigned long*) 0x4000C02C))
#define GPIO_PA_DEN (* ((volatile unsigned long*) 0x4000451C))
#define GPIOPCTL (* ((volatile unsigned long*) 0x4000452C))
#define GPIOAMSEL (* ((volatile unsigned long*) 0x40004528))
#define UARTCC (* ((volatile unsigned long*) 0x4000CFC8))
```

```
#define UARTIM (* ((volatile unsigned long*) 0x4000C038))
#define UARTDR (* ((volatile unsigned long*) 0x4000D000))
#define UARTICR (* ((volatile unsigned long*) 0x4000C044))
#define UARTFR (* ((volatile unsigned long*) 0x4000D018))
#define DMACHIS (* ((volatile unsigned long*) 0x400FF504))
#define UARTRIS (* ((volatile unsigned long*) 0x4000C03C))
#define GPIO_PCTL_PORTB     (*(volatile uint32_t *)0x4000552C)
#define UART_CTL            (*(volatile uint32_t *)0x4000D030)
#define UART_IBRD           (*(volatile uint32_t *)0x4000D024)
#define UART_FBRD           (*(volatile uint32_t *)0x4000D028)
#define UART_LCRH           (*(volatile uint32_t *)0x4000D02C)
#define UART_CC             (*(volatile uint32_t *)0x4000DFC8)

#endif
```