Engineering Portfolio

Peter Ly

Profile

Name: Peter Ly

Major: Computer Engineering

Email: peterly0516@icloud.com

Github: https://github.com/peterly-star?tab=repositories

LinkedIn: www.linkedin.com/in/peter-ly-4553b4374

About this Portfolio:

The goal of this portfolio is to highlight several projects I have worked on throughout my time in college. These projects primarily focus on embedded systems, microcontroller-based design, and the integration of hardware and software. Through these projects, I have gained hands-on experience in areas such as sensor interfacing, motor control, real-time programming, and system debugging.

Project timeline

Line Following Robot W/O Microcontrollers

Page 4

Line Following Robot With Microcontrollers

Page 6

Object/Wall Following Robot

Page 9

Space Invaders Game

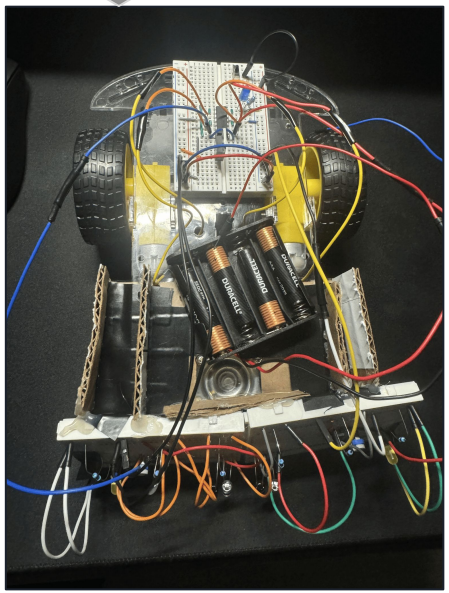
Page 12

Line Following Robot W/O Microcontrollers

Overview

- This was a final project for my CECS 262 course where we had to design and build a line following robot that uses no type of microcontroller or arduinos
- The goal was to build a line following robot that follows in between a black tape line around a track course

Final design for line following robot



- Track course used for line following robot testing
- Straight line and sharp turns to test the robot's movements

Video Link

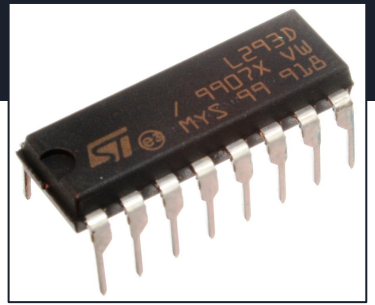
Contributions:

- The project was worked on with another partner as it was part of the final project to work in pairs
- Worked on breadboarding the L239D motor driver/functions of the driver, power switch for the car to turn on and off, and car mounting design for the sensors

Challenges:

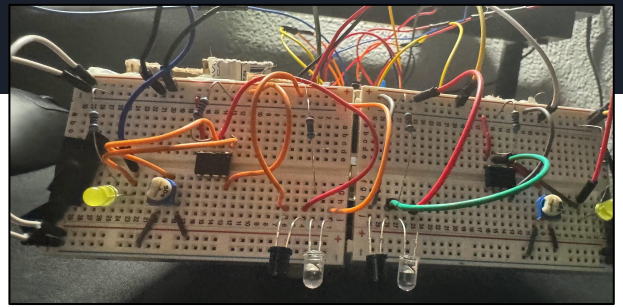
- Deciding whether we wanted to try to do soldering connections on a pcb board for a cleaner look but we ended up settling to wire it on breadboards
- Figuring out the range and sensitivity of how light was being read from the sensor in order for the robot to follow the line smoothly

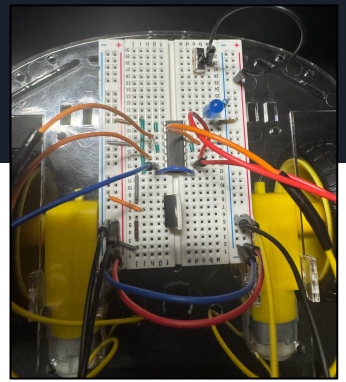
L293D Motor Driver



Assembly car kit frame

IR Sensors breadboard circuit





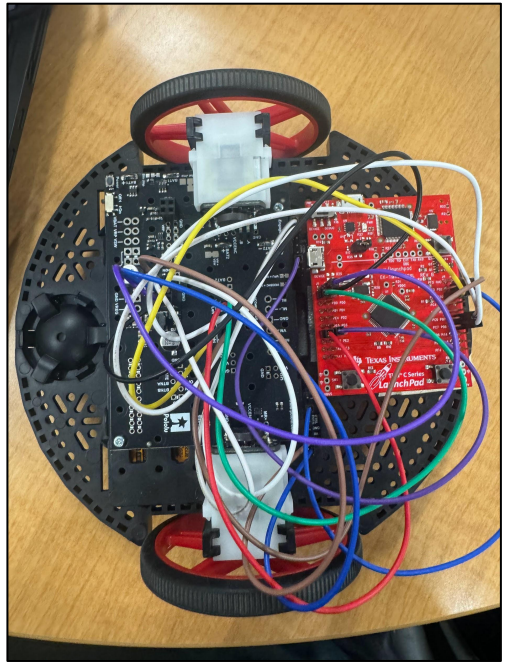
Motor Driver breadboard circuit

Line Following Robot With Microcontrollers

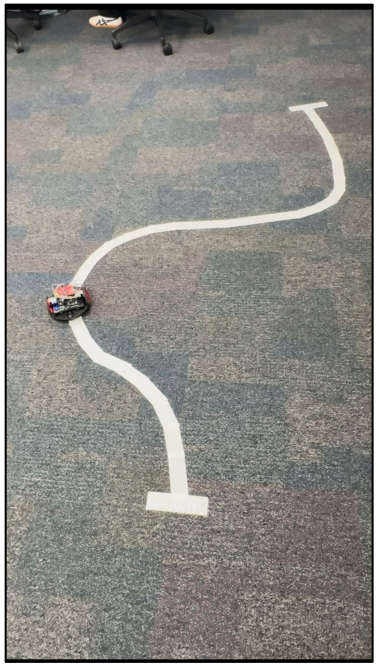
Overview

- This was a final project for my CECS 346 course where we had to develop a line following using a TM4C123 board and a romi chassis car robot that was provided
- The goal was to develop a line following robot that follows in between a white tape line through the embedded system program Keil using C language

Final design for line following robot



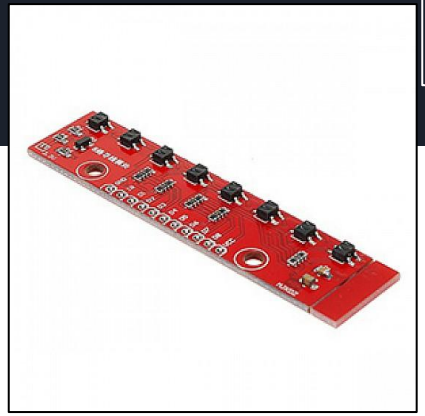
One of the track courses used for line following robot testing

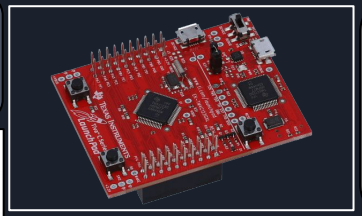


Contributions:

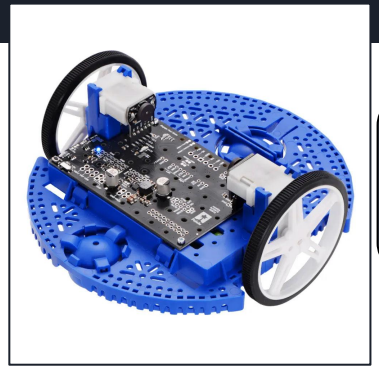
- The project was done in a group of four, each of us had assigned parts
- Calibrated which channels to use in the Keil program for the QRT Reflectance IR sensor so the robot can following the white line smoothly
- Worked on the pin connections on the TM4C123 board with the car chassis and sensor

QRT Reflectance IR sensor (8 different sensor channels)





TM4C123 board used for sensor/car connections



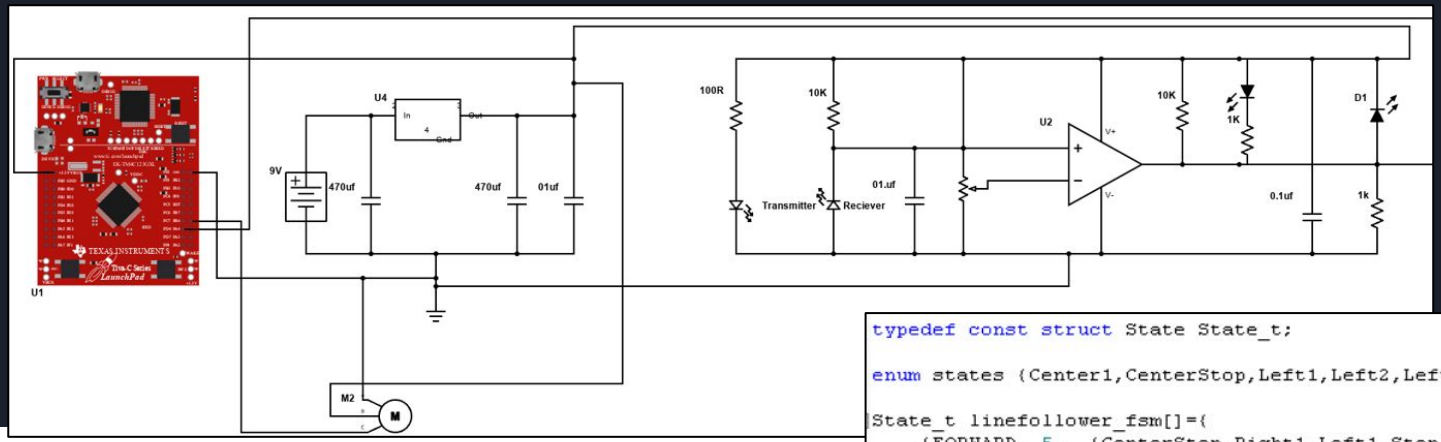
Romi chassis robot frame given

Below is the code used to declare sensor channels and timing outputs when being read

```
#define SENSOR_CTRL_PINS     0x03  // PB0 = CTRL EVEN, PB1 = CTRL ODD
#define SENSOR_PINS          0x03  // PE0 = Sensor 0, PE1 = Sensor 7

// This function performs a sensor read by charging and measuring reflectance sensors.
uint8_t Sensor_CollectData(void) {
    uint8_t sensor_data;

    // Disable IR LEDs (turn off emitters)
    SENSOR_CTRL &= ~SENSOR_CTRL_PINS;
    Wait_N_MS(2);

    // Enable IR LEDs (turn on emitters)
    SENSOR_CTRL |= SENSOR_CTRL_PINS;

    // Configure sensor pins as output and drive high to charge capacitors
    GPIO_PORTE_DIR_R |= SENSOR_PINS;      // Set PE0 and PE1 as outputs
    SENSORS |= SENSOR_PINS;               // Output HIGH on PE0 and PE1

    Wait_N_US(10);                        // Allow capacitors to charge for 10us

    // Configure sensor pins as input to measure discharge
    GPIO_PORTE_DIR_R &= ~SENSOR_PINS;     // Set PE0 and PE1 as inputs

    Wait_N_US(460);                       // Wait for discharge time

    // Read reflectance sensor values (higher value = more reflectance)
    sensor_data = SENSORS;

    // Turn off IR LEDs
    SENSOR_CTRL &= ~SENSOR_CTRL_PINS;
    Wait_N_MS(10);

    return sensor_data;
}
```

Challenges:

- Figuring out the state algorithm for the robot to have it smooth turns and movements
- Having the IR sensor work with the robot motors as in some cases the the motor was moving nonstop or sometimes not detecting white/black leading to movement errors



Final Schematic design for line following robot

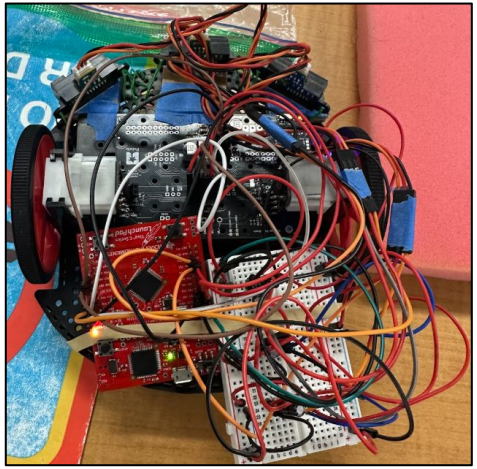
Figure to the right shows the states that are called for each movement period along with a value that determines the car motor timing/speed when the car stops, move or turns

```
typedef const struct State State_t;

enum states {Center1,CenterStop,Left1,Left2,LeftStop,Right1,Right2,RightStop,Stop};

State_t linefollower_fsm[]={
    {FORWARD, 5,  {CenterStop,Right1,Left1,Stop} },    //Center
    {FORWARDSTOP, 150,  {Center1,Right1,Left1,Stop} },    //Center

    {TURN_LEFT,    3,  {Center1,Left2,Right1,Stop} },   //LEFT fastest
    {LEFTSTOP,   250,  {Center1,LeftStop,Right1,Stop} },   //LEFT medium speed
    {TURN_LEFT,    2,  {Center1,Left1,Right1,Stop} },   //LEFT slow

    {TURN_RIGHT,  3,  {Center1,Left1,Right2,Stop} },   //RIGHT
    {RIGHTSTOP,  250,  {Center1,Left1,RightStop,Stop} },   //RIGHT
    {TURN_RIGHT,  2,  {Center1,Left1,Right1,Stop} },   //RIGHT

    {STOP,   40,  {Center1,Left1,Right2,Stop} }     //STOP
  };
```

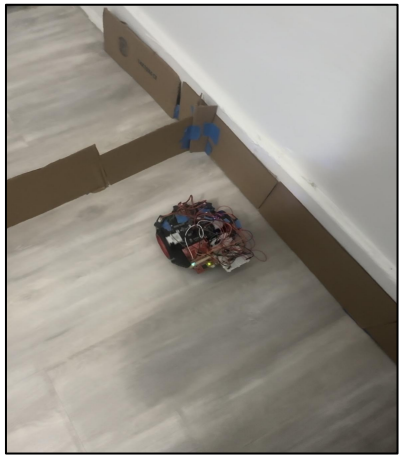
Object/Wall Following Robot

Overview

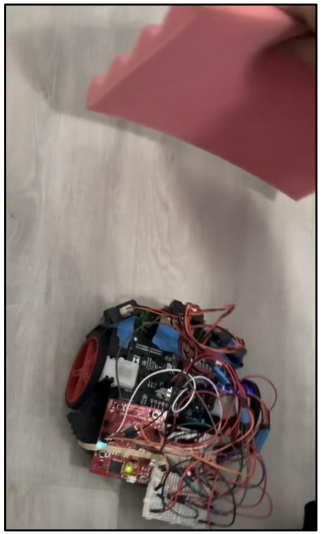
- This was a project for my CECS 347 course where we had to design and build a object/wall following robot using TM4C123 board, IR Distance Sensors, and previous knowledge built from line following robot
- The goal was to develop the car robot with two modes in the Keil program; mode 1 being a object following robot that follows an object at a certain distance and then mode 2 having the robot to follow along a wall



Left image is final design of the robot for both modes



Wall Follower
Video Link



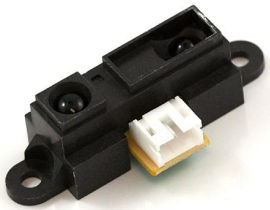
Object follower
Video Link

Robot following an object in the image above and robot following along a wall on the left image

Contributions:

- The project was done in a group of four, each of us had assigned parts
- Worked on the physical pin connections of the car robot and building the breadboard circuit for the IR Distance Sensors
- Recorded readings from the IR Distance Sensors and used that data to code into the Keil program to make sensors work with the robot

IR Distance Sensors: Light reflects off the object and gives distance value of the object (ADC)



Sensor 1:

Data Sets	15cm	20cm	30cm	70cm
1	2370	1665	1152	567
2	2362	1642	1160	579
3	2365	1708	1172	546
4	2346	1698	1157	595
5	2350	1710	1162	597
Average	2358.6	1684.6	1160.6	576.8

- Sample of a data set for one of the sensors above
- Recorded the sensor value at four different distances five times
- Took the mean of each distance and used it as a reference value to use for car movements at different distances

Code shown below is getting multiple value readings from each IR Distance Sensors and taking the median value to use

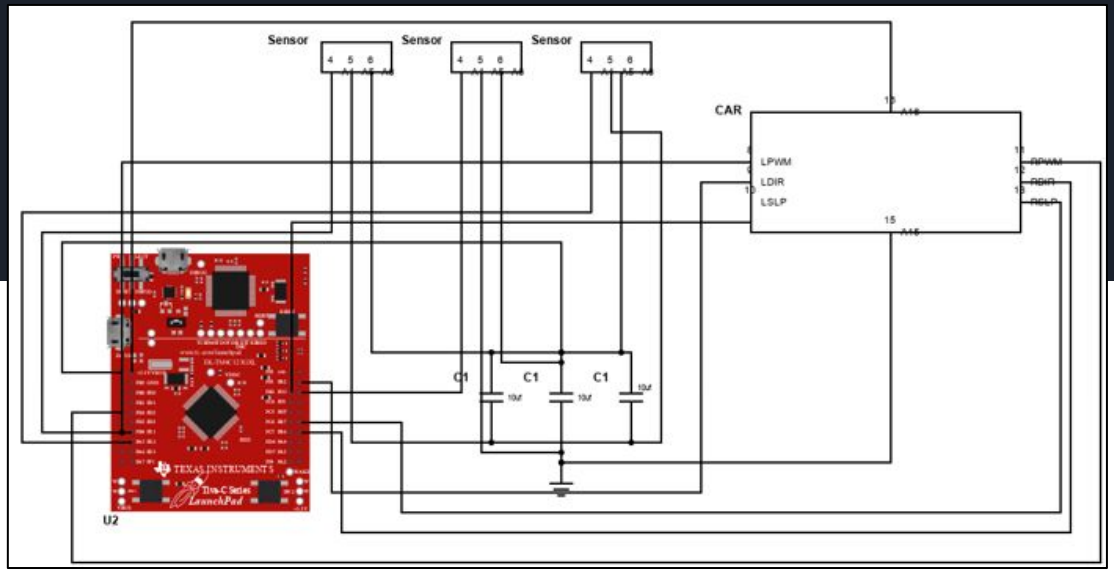
```
// Read all three sensors with median filtering
void ADC1_ReadAllSensorsFiltered(uint16_t *left, uint16_t *front, uint16_t *right)
{
    // Static variables to keep history for median filtering
    static uint16_t left_oldest = 0, left_middle = 0;
    static uint16_t front_oldest = 0, front_middle = 0;
    static uint16_t right_oldest = 0, right_middle = 0;

    uint16_t left_newest, front_newest, right_newest;

    // Read newest values
    ADC1_InSS1(&left_newest, &front_newest, &right_newest);

    // Calculate median for each sensor
    *left = median(left_newest, left_middle, left_oldest);
    *front = median(front_newest, front_middle, front_oldest);
    *right = median(right_newest, right_middle, right_oldest);

    // Update history
    left_oldest = left_middle;
    left_middle = left_newest;

    front_oldest = front_middle;
    front_middle = front_newest;

    right_oldest = right_middle;
    right_middle = right_newest;
}
```

Challenges:

- Each IR Distance Sensor were reading very different from one another so we had to have different values for each distance used from each sensor (shown in figure on the bottom left)
- Mounting each sensor to the car was a bit tricky as each of the sensor needs to be separated at the same height and distance from each other or the values would be off from the code

As you can see, value for FRONT_20_ADC is different than the LEFT/RIGHT_20_ADC

```
// ======= Distance thresholds
// Front thresholds
#define FRONT_20_ADC        1660
#define FRONT_20_BAND       90
#define FRONT_10_ADC        2495
#define FRONT_10_BAND       90
#define FRONT_70_ADC        615
#define FRONT_70_BAND       50

// Left thresholds
#define LEFT_20_ADC         1800
#define LEFT_20_BAND        100
#define LEFT_10_ADC         2600
#define LEFT_10_BAND        60
#define LEFT_70_ADC         620
#define LEFT_70_BAND        40

// Right thresholds
#define RIGHT_20_ADC        1550
#define RIGHT_20_BAND       60
#define RIGHT_10_ADC        2425
#define RIGHT_10_BAND       60
#define RIGHT_70_ADC        680
#define RIGHT_70_BAND       40
```

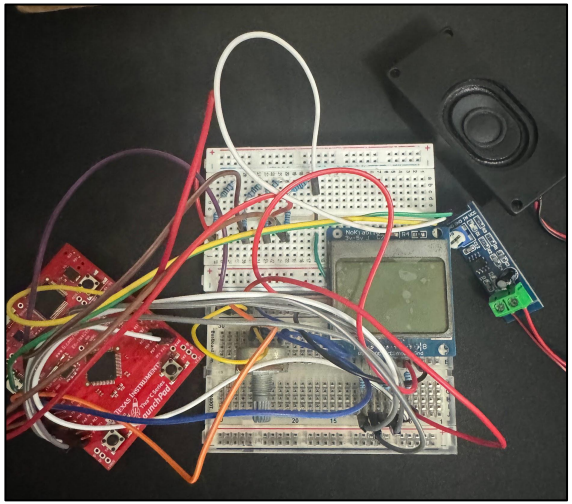


Final Schematic design for object/wall following robot: capacitors were used to make sure values weren't being affected by other objects

Space Invaders Game

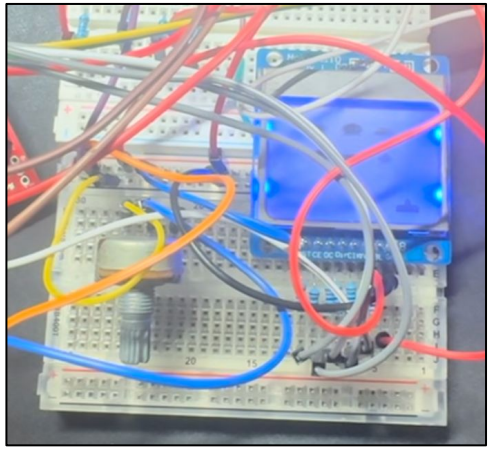
Overview

- This was the final project for my CECS 347 course where we had to develop a real-time interactive game on the TM4C123 board
- The goal was to code a space invaders game in the Keil program and use the TM4C123 board to display the game on a Nokia 5110 LCD
- Implements player input such as shooting/movements, enemies moving across the screen and outputs game sounds through a speaker



Final breadboard circuit of Space Invaders game implemented on a Nokia 5110

When game starts, it displays enemies at the top and the player at the bottom of the LCD

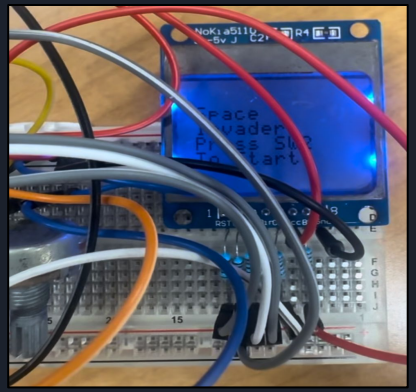


Space Invaders Game Video Link

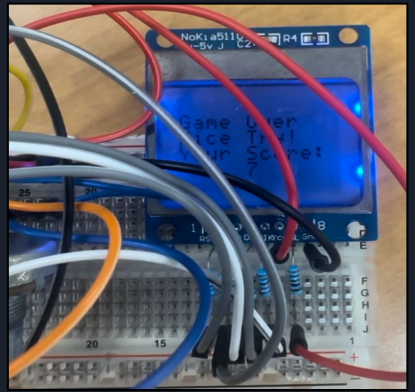
Contributions:

- The project was done in a group of four, each of us had assigned parts
- Worked on the wiring connections for speakers and Nokia 5110 LCD screen
- Coded game starting/end screen, player movements, and sound of the game

Starting screen for the game displaying “Space Invaders Press SW2 To Start”



End screen for the game displaying “Game Over Nice Try! Your Score: X”



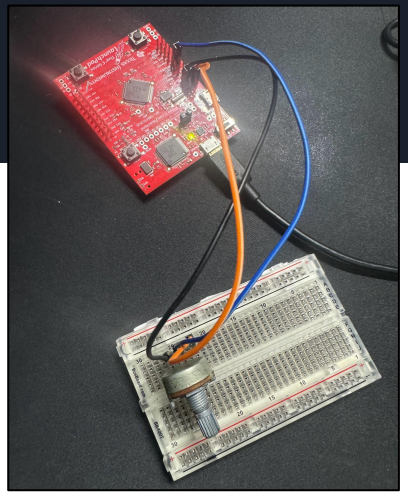


Figure on the left shows early developments of the project where I was working on the player movements using the potentiometer on the breadboard connected to the TM4C123

For both images below, the code clears all the text on the Nokia, then writes out the text at each screen bit position from the top to bottom during start/end of game

```
//Starting text
void Start_Prompt(void) {
  Nokia5110_Clear();
  Nokia5110_SetCursor(0, 1);
  Nokia5110_OutString("Space");
  Nokia5110_SetCursor(0, 2);
  Nokia5110_OutString("Invader");
  Nokia5110_SetCursor(0, 3);
  Nokia5110_OutString("Press SW2");
  Nokia5110_SetCursor(0, 4);
  Nokia5110_OutString("To Start");
}
```

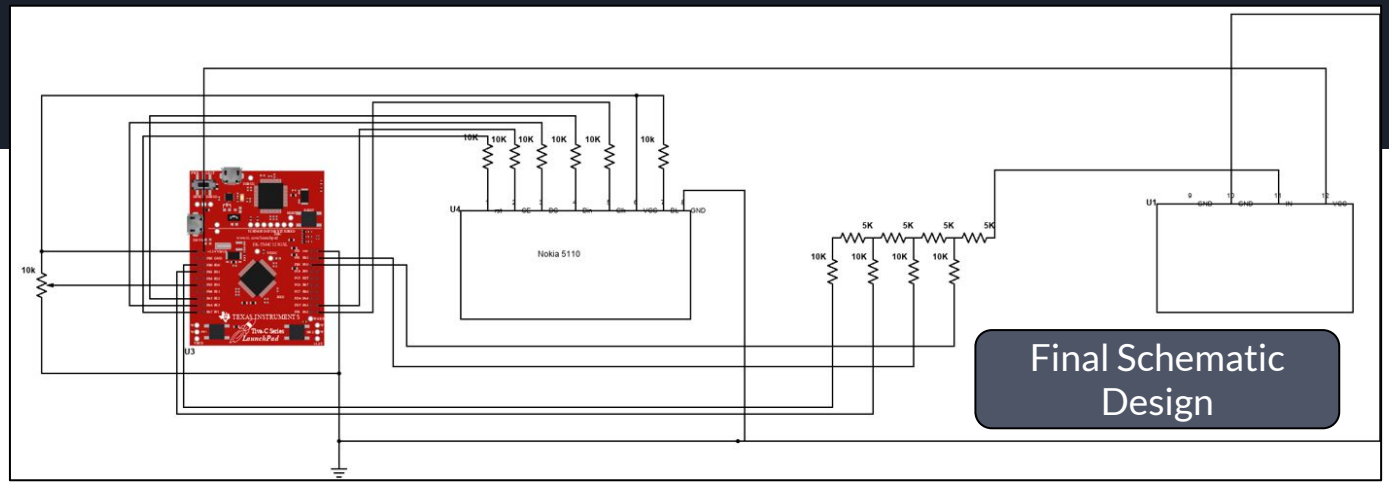
```
//End of game score
void End_Prompt(void) {
  char scoreStr[5];
  sprintf(scoreStr, "%d", score);  //

  Nokia5110_Clear();
  Nokia5110_SetCursor(0, 1);
  Nokia5110_OutString("Game Over");
  Nokia5110_SetCursor(0, 2);
  Nokia5110_OutString("Nice Try!");
  Nokia5110_SetCursor(0, 3);
  Nokia5110_OutString("Your Score:");
  Nokia5110_SetCursor(5, 4);
  Nokia5110_OutString(scoreStr);
```

Challenges:

- Configuring the audio amplifier to output actual sounds from the game to the speakers as it sometimes output frequencies from actual radio channels
- Coding the enemies into the Nokia screen, the enemies would appear out of the screen or in the middle of the screen when it was needed at the left side moving towards the right

Image below shows coding of the sound that was implemented to the game, sound is played based on timing of player actions, such as shooting a bullet

```
void Sound_Play(const uint8_t *pt, uint16_t count){
  if (count == 0) return;

  Wave  = pt;
  Index = 0;
  Count = count;

  TIMER1_ICR_R = 0x00000001;                  // clear timeout flag
  TIMER1_TAILR_R = 80000000/11025 - 1;        // (re)set period in case it changed
  TIMER1_IMR_R |= 0x00000001;                 // arm interrupt
  TIMER1_CTL_R |= 0x00000001;                 // enable TIMER1A
}

// Simple wrappers for your existing wavetable data
void Sound_Shoot(void){
  Sound_Play(shoot, 4080);
}
```



Final Schematic Design

LM386 Audio Amplifier, connects to a speaker to convert small audio signals and amplifies the audio sound