Final Report: The Fire truck Project

Samuel Kleiner, Richard Ferrer-Westrop

December 2023

1 Implementing PID Flame Detection and Destruction

1.1 Statement of Objective

Implementing PID Flame Detection and Destruction [FDD] for our car required the designing, building, and testing of a servo and fan control board, and a new power board. It required the purchasing and implementation of PWM-controlled fans, servos, and speakers, and the integration of an Arduino Mega board into our system. We also worked with a Liquid Crystal Display (LCD), TR-EVO-T33-UART thermal camera, and had to print 11 different mounts designed in CAD, and one designed in CorelDraw that was laser cut, to integrate all of our new hardware into our system. Finally the project required the programming of the PSoC's firmware and software to implement the FDD algorithm and control on our PSoC, and the programming of the Arduino Mega to run the Speaker System, process the thermal camera's output, and run the LCD system.

Our goal over the last month has been to have our car detect a flame using a thermal camera, navigate to the flame, and blow the flame out with a fan mounted on the car. First, I sketched out the design for the Fire truck. We would start with one fan on one servo, a thermal camera, and an Arduino to process the output of the thermal camera. Then, I added a speaker system to the design, another fan to improve the range of flame destruction capability, and a servo for the thermal camera to increase the range of flame detection. Finally, a LCD was ordered to output the image from the thermal camera, or Fire truck related text and imagery. Mounts for each part were sketched, then designed, and finally printed. Using the data sheets from the fans, and the available information on the servos, I tested each component as we received them until we understood the how the fans and servos were controlled by PWMs. I designed a circuit board to provide the two fans and three servos the necessary power, ground, and PWM signals. I had Rich solder a new power board to provide +5 V and ground for the fan and servo board. We mounted all the components. I wrote the code to send the appropriate PWMs to the fans and servos from the PSoC, and the PWMs from the Arduino Mega to create the siren from the

speakers. The testing of our thermal camera and LCD system will be discussed in the section on Design Choices and Challenges.

Testing the fans, servos, and related subsystems of the car was critical to my successful demonstration of the Fire truck's capabilities. The Oscilloscope and DC Power Supply was used extensively to test the thermal camera's signals and connections, and the firmware and software implemented on the PSoC by routing signals to two output pins on the PSoC.

I would like to thank the above and beyond efforts of the Professors, Graduate TAs, and Undergraduate TAs in helping me implement the Fire truck's flame detection and destruction capabilities. I could not have understood, completed, and debugged my designs, circuit boards, or code, without your help. I am extremely grateful, and want to especially thank the efforts of Michael and the Professors in helping me work with the thermal camera.

1.2 Key Sub-systems and Components

Here I provide diagrams and photos of the key sub-systems and components of my car that have been changed or created since the beginning of the Fire truck project.

1.2.1 Fan and Servo board

The soldered board and design of our fan and servo board is shown below. Everything is connected with a KK connector. The connections to the two fans and three servos each provide +5 V, ground, and a specific PWM from the PSoC. This schematic includes the two fan connections, three servo connections, connection to the PSoC, and connection to our second power board.

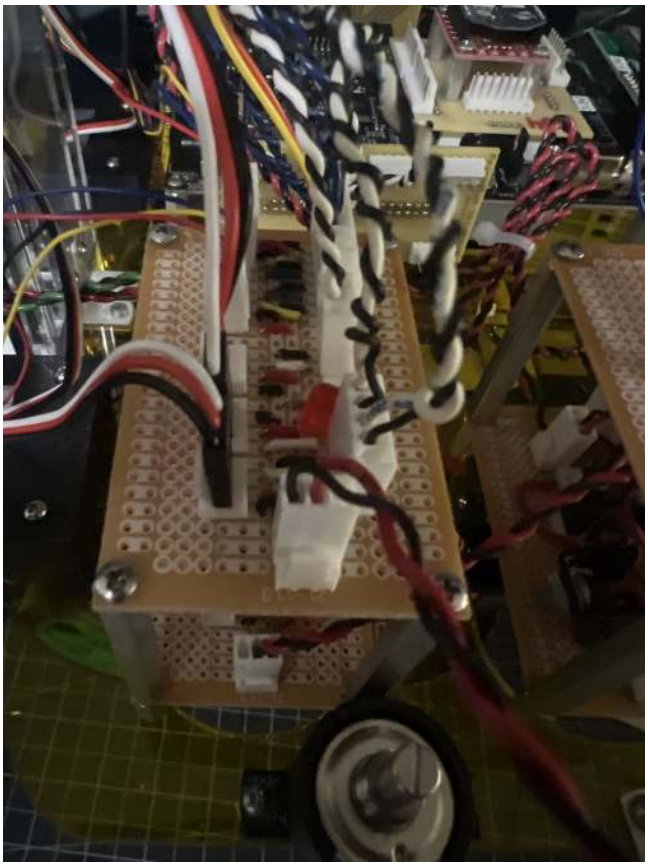


Figure 1: Fan and Servo board

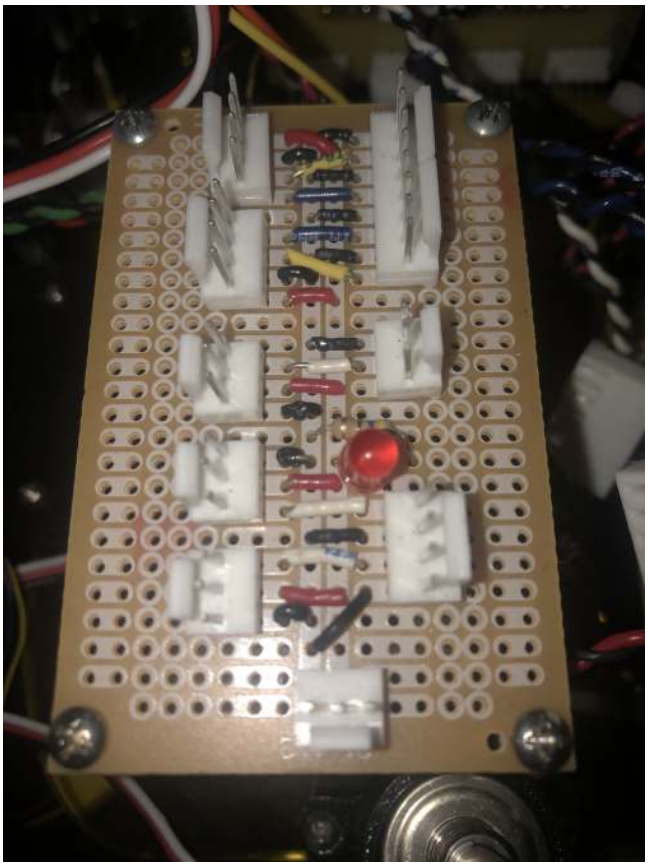


Figure 2: Fan and Servo board with connections removed

1.2.2 Second Power board

We had already altered our original power board. There was not room for a new, clean connection, so we made a second power board. It was mounted on standoffs above the first power board.

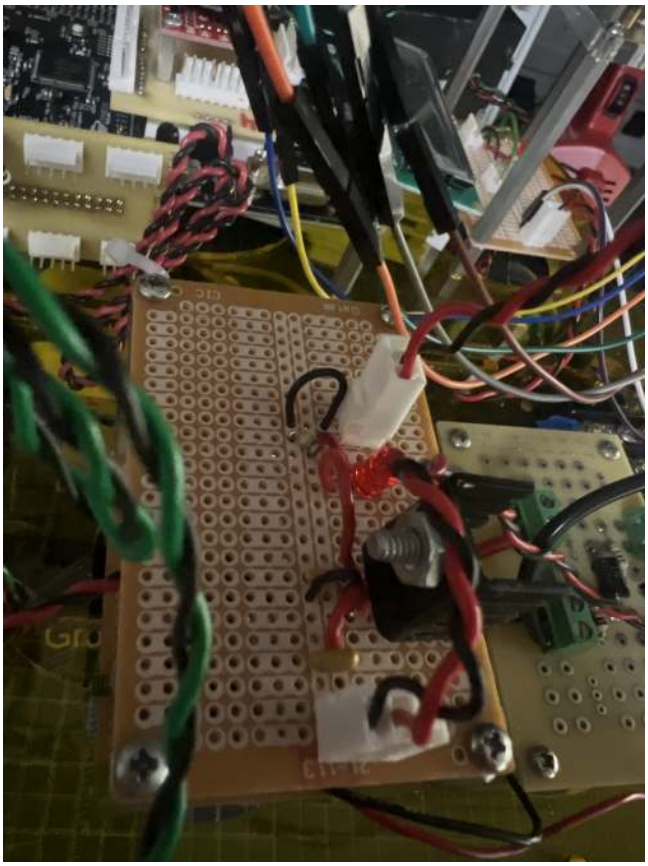


Figure 3: Second Power board

1.2.3 Ardunio Mega

We used an Arduino Mega to run the Speaker system, and attempt to run the LCD system and thermal camera system. We used Serial to debug the Arduino's code, which is displayed in the Appendix. We used Serial1 in our communication with the thermal camera, and Serial3 to communicate with the PSoC. The Arduino Mega was mounted on the back of the Fire truck's mast, so its central location would allow easy access for more secure connections.

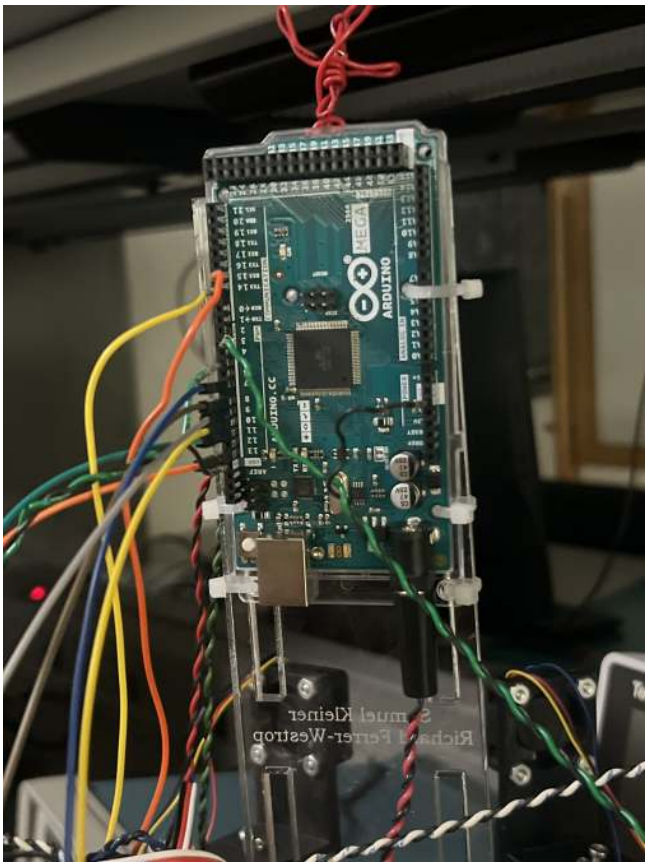


Figure 4: Our Arduino Mega mounted to the Fire truck

1.2.4 Speaker System

Our speaker system was comprised of two powerful speakers each with a potentiometer that controlled their volume. The speakers were run off a PWM from our Arduino Mega, which I programmed to sound like an emergency siren.

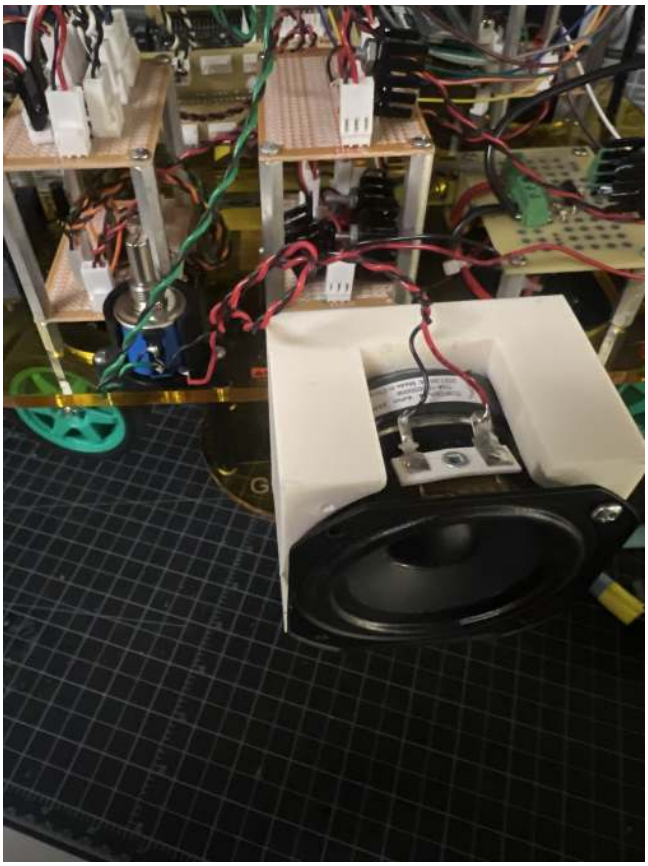


Figure 5: Speaker mounted to the Fire truck. Volume control can be seen to the left

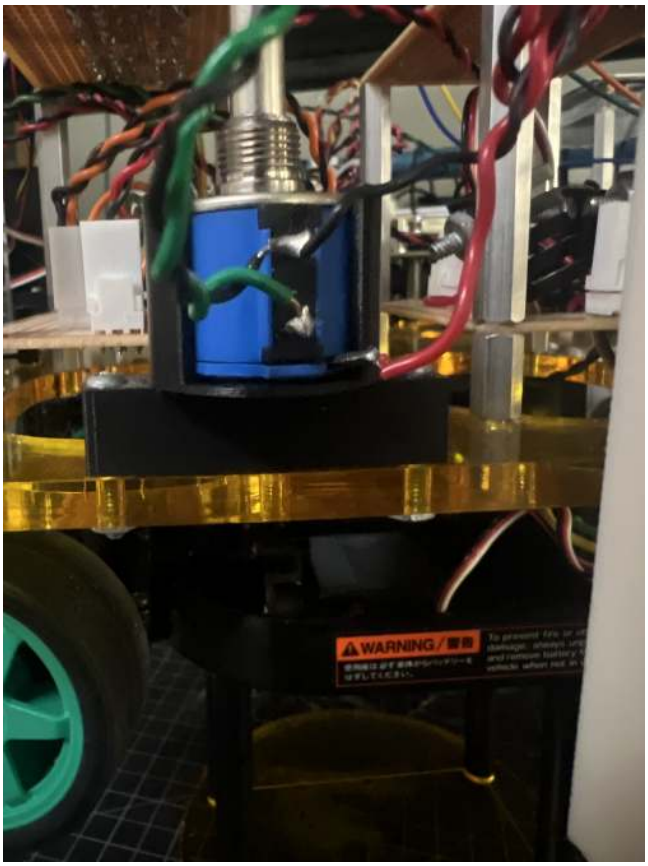


Figure 6: Volume control

1.2.5 Fan and Servo System

Our fan and servo system was comprised of two fans each mounted to a servo mounted to the Fire truck. The white paper in the background blocks the black color of the fans, servos, and their mounts from the C-Cam-2A camera that was mounted on top of our mast for Demo day.

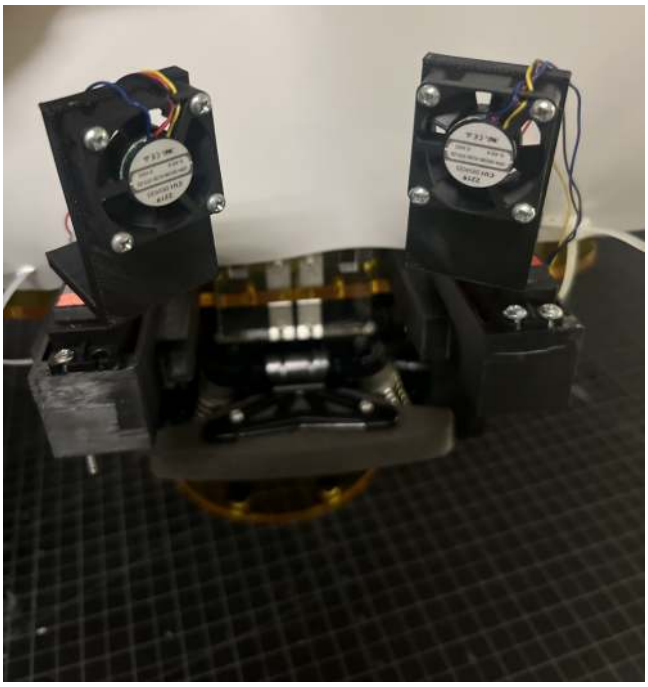


Figure 7: Fans mounted on Servos mounted to Car

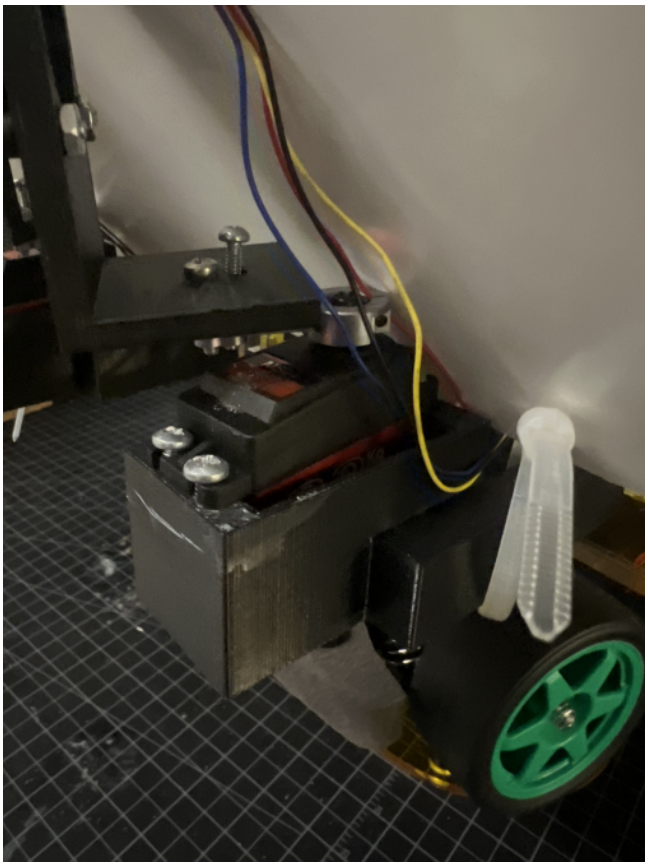


Figure 8: Side view of Fan mounting system

1.2.6 Thermal Camera System

While the thermal camera system did not make it into our final design for Demo day, here is the system we designed, mounted, and could use in the future to improve our fire detection and destruction capabilities. The TR-EVO-T33-UART thermal camera was mounted to a servo. The servo was mounted to our Fire truck's mast. This allowed the thermal camera to be connected to the Arduino, which was mounted on the back of the mast, and have a clear 270view from the front of the Fire truck for fire detection.

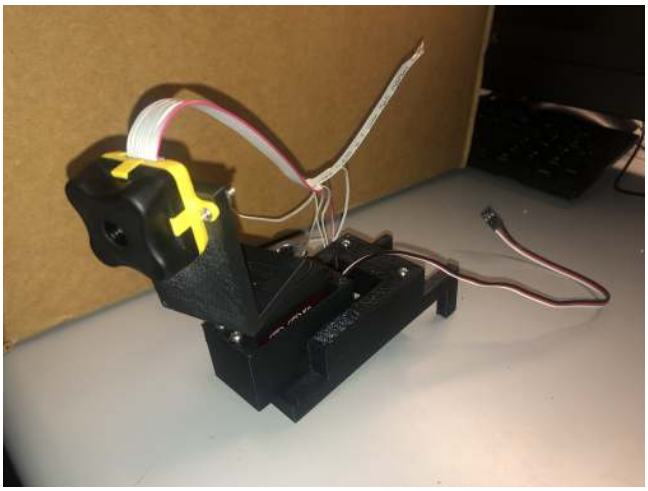


Figure 9: Thermal Camera with mount

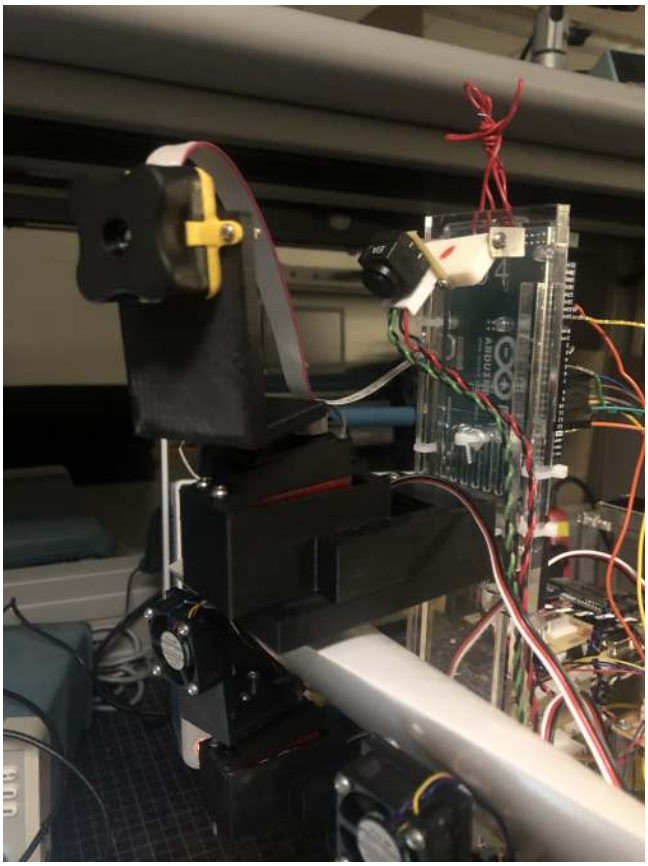


Figure 10: Thermal Camera on Fire truck

1.2.7 LCD System

Our LCD display was mounted on a laser cut piece of a acrylic with the same dimensions as our circuit boards, so that it could be mounted on the Fire truck on top of our Hall Effect board using standoffs. We laser cut four more holes in the acrylic that fit the dimensions of the standoffs that came with the two inch LCD. The LCD was mostly connected to the Arduino Mega, but received its +5 V and ground from the Hall Effect board since it was no longer being used to power the Hall Effect sensor.

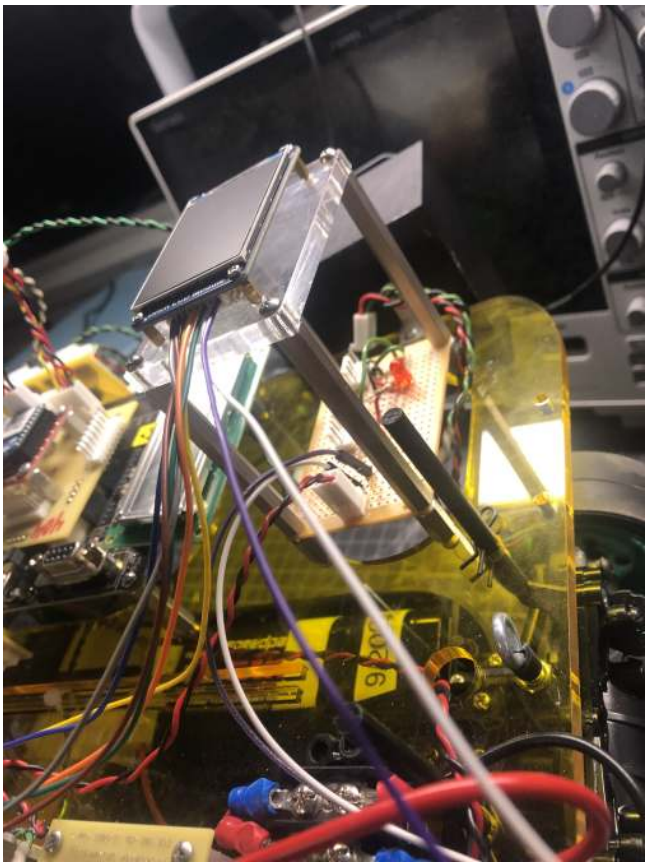


Figure 11: LCD system mounted on the board

1.2.8 PSoC firmware and hardware

The firmware I implemented for the Fire truck project in our TopDesign.cysch and Design01.cydwr files using the PSoC Creator 4.4 IDE are shown in our Appendix. The PSoC hardware worked to implement the firmware and run the software successfully.

1.3 Data from Testing

I did twenty trials excluding a successful one on demo day, starting with charged batteries. The Fire truck succeeded in successfully demonstrating its fire destruction capabilities by blowing out a candle mounted at the height of the fans in all but two of the trials. In those two trials, the Fire truck deviated slightly from the black line on the floor, and the fans did not end up pointing at the candle. Both trials where failure occurred were near the end of the twenty trials (trials 16 and 19). I noticed a decrease in the reliability of the navigation and

performance of the Fire truck in the testing leading up to Demo day, which was correlated to the charge of the battery. Starting with a fully charged battery led to a more reliable performance. A simple fix to correct for the failure in trials 16 and 19 would have been to do a 270 degree sweep with each fan's servo since the candle was well within range of the fans, but the fans were not pointed at the candle.

1.4 Design Choices and Challenges

The first significant design choice after sketching out the Fire truck's basic design and ordering the parts was deciding which parts would be controlled by the Arduino and which would be controlled by the PSoC. After some deliberation, I decided to use the PSoC to provide the PWMs for the fans and the servos, and to have the PSoC communicate with the Arduino, which would run the thermal camera image processing, speaker system, and LCD system.

The first challenge we faced was a delay in receiving the thermal camera and LCD system since they were ordered later than they should have been. The USB backboard for the thermal camera was delivered very late for the same reason. Receiving that increased our abilities to debug the camera's output because it could be displayed on a computer with a GUI provided by TeraBee. This challenge was a challenge of our own making.

The LCD was received so late that attempts to have it produce Fire truck related imagery and text in the days before our demonstration were unsuccessful when a few hours of working with the provided sample codes were unsuccessful.

We also experienced a significant challenge in producing usable data from the TR-EVO-T33-UART thermal camera, and an image from the LCD. Hours of debugging and help from Professor Valavi led to the discovery that the Tx and Rx pins on the first thermal camera we were working with were broken. We had ordered an extra since it was a critical part of our Fire truck's system, so that problem was solved. We were initially working with an Arduino Uno, but I realized that the memory constraints made it impossible to process the video frames from the thermal camera, which were each 2070 bytes. However, even working with the second thermal camera on our Fire truck, an Arduino Mega, and using the first one with a USB backboard to help debug in the final days before Demo day, lead to no useful data from the thermal camera. Using the TR-EVO-T33-UART's User manual, all available specifications and information on the data sheet, and GitHub with sample codes was not enough to produce useful data by the morning of Demo day.

I realized that the GitHub provided sample code for a variety of sensors from TeraBee, yet not the thermal camera we had. I was able to adapt the sample code for a distance sensor provided by TeraBee (the TeraRanger-EVO-Mini), to work with our thermal camera. A depiction of the thermal camera's signal

structure is below.

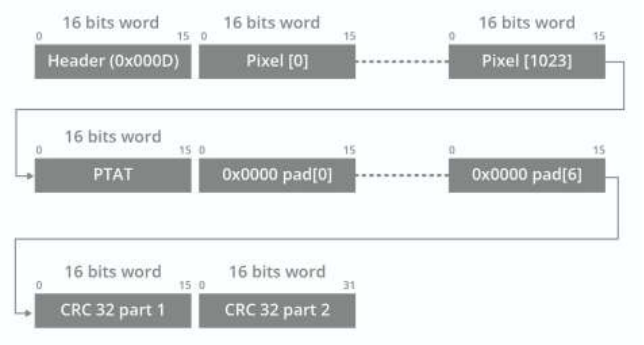


Figure 12: Evo Thermal output format structure

The TeraRanger Evo Thermal User Manual provided the thermal camera's output format, but its documentation needs to be improved. The thermal camera was deduced to use a CRC32 checksum, but the manual provides code for a CRC8 checksum, which was confusing.

Various attempts were made to produce useful data from the thermal camera after the first byte of each frame (decimal 13, the first byte of a two byte header) was consistently identified at the start of each frame the Arduino Mega received using the Serial Monitor, but every other 16 bit word that followed the first byte seemed arbitrary.

Professor Lyons suggested making no assumptions about its output, and creating a half hot, half cold image for the thermal camera to see. I created that set up using a hot coffee cup and a room-temperature wall. The thermal camera's image was confirmed using the second thermal camera with a USB backboard, but the first thermal camera's output was still seemingly random. Other attempts to debug the thermal camera's output using the Arduino Mega's Serial port were unsuccessful. After discussing the problem with Michael in the early hours of Demo day, I decided to go with Plan B.

Plan B involved reintegrating the C-Cam-2A that was used for PID Navigation Control onto our Fire truck, manually coding the location of the flame, and optimizing for maximum reliable flame destruction in the time I had before my demonstration. I created a timer on the PSoC to run the Fire truck's motors for approximately two seconds. I measured how far the Fire truck would travel along the black line of tape on the ground from our Navigation phase of Carlab. I created a mount for the candle, wrapped it in black tape, and put the candle where I knew the Fire truck would stop. I programmed the Fire truck to move the servos that the fans were mounted on to the position that I placed the candle after the Fire truck had arrived at the candle.

Testing of Plan B showed me that I had to improve the PID Navigation control algorithm that I had used to pass Navigation, so that the Fire truck more reliable stayed precisely on the black line, and its fans ended pointed at the candle. I added and tuned a derivative term to my PID Navigation control algorithm. While Plan B demonstrated a far less impressive control algorithm for flame detection and destruction, it did lead to a successful demonstration on Demo day at our demonstration time that I was proud of. It demonstrated the integration of a number of different components and systems, and taught me valuable lessons along the way.

We did a pretty good job of measuring twice, and 3D printing once, but we still had to modify our prints and acrylic LCD mount with the drill press and Professor Lyons help, which taught me to measure three times, and continue to improve communication with my partner. I also learned that the most critical subsystem of any project must be taken on first. While there were delays with receiving the thermal camera, I completed the designing, programming, and implementation of all other less critical subsystems of the Fire truck, before the thermal camera. This did result in Plan B being possible to be implemented on Demo day, but, if I had worked more extensively on the thermal camera system earlier, I might have been able to order another thermal camera, or figure out another solution.

1.5 Future Improvements

Our greatest challenge created our greatest opportunity for future improvement: the successful implementation of a thermal camera system into the Fire truck. Additionally, I would have liked to increase communication between the PSoC and Arduino. The Serial ports could be used so that the speaker stops doing the siren after a certain amount of time, or turn off once the flame was out.

1.6 Appendix

1.6.1 Updated PSoC firmware and hardware

TopDesign.cysch

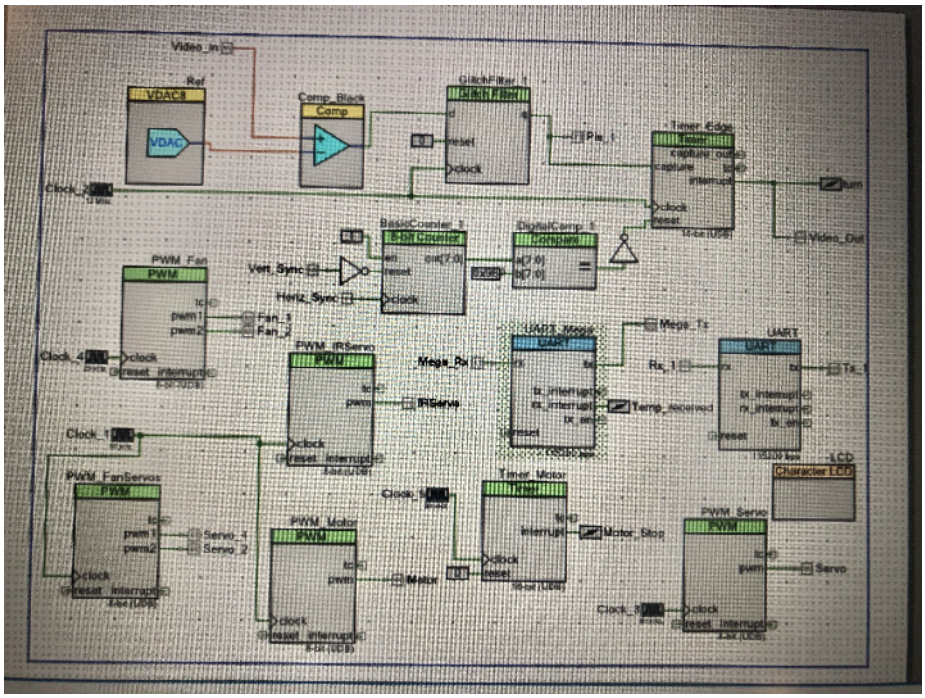


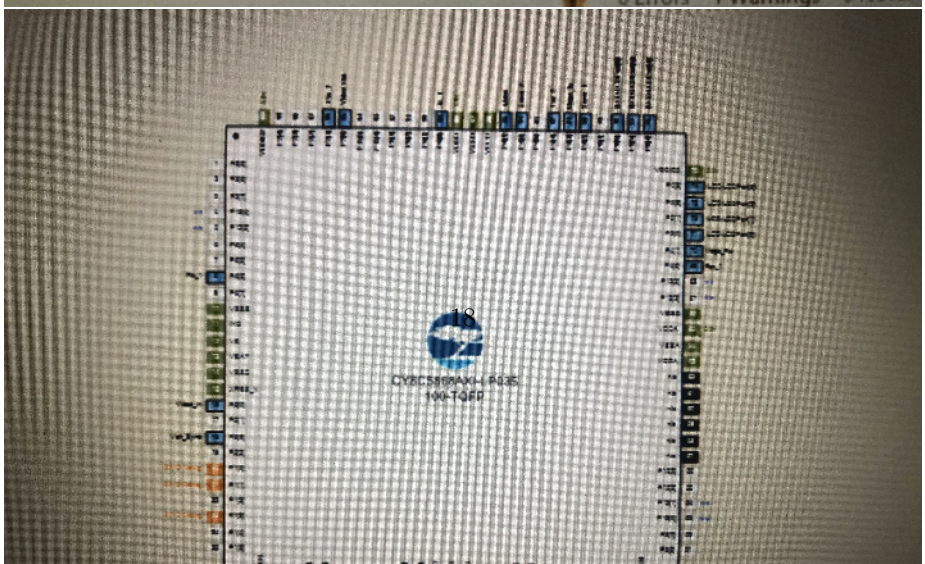
Figure 13: PSoC firmware

Design01.cydwr

	Name	Port	Pin	Lock
	\LCD:LCDPort[6:0]\	P0[6:0]	78...76,74...71	☑
	Fan_1	P4[0]	69	☑
	Fan_2	P4[4]	82	☑
	Horiz_Sync	P5[4]	31	☑
	IRServo	P1[6]	27	☑
	Mega_Rx	P4[1]	70	☑
	Mega_Tx	P4[3]	81	☑
	Motor	P4[7]	85	☑
	Pin_1	P2[1]	96	☑
	Rx_1	P6[6]	8	☑
	Servo	P5[6]	33	☑
	Servo_1	P4[2]	80	☑
	Servo_2	P4[6]	84	☑
	Tx_1	P6[0]	89	☑
	Vert_Sync	P5[2]	18	☑
	Video_in	P5[0]	16	☑
	Video_Out	P2[0]	95	☑

EEPROM

0 Errors 1 Warnings 0 Notes



1.6.2 PSoC Code (main.c)

```
  /* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include "project.h"
#include <stdio.h>
#include <math.h>

uint16 old_time = 65535;
uint16 new_time = 0;
uint16 dt = 0;
double oldspeed = 0;
double speed = 0;
double target = 4;
double n0 = 50;
double n = 50;
double error = 0;
double integration = 0;
double derivative = 0;
double Kp = 80;
double Ki = 1;
double Kd = 0;
double diameter = .216535;

double percent;
char strbuf2[16];
double PWM_Steer;
double kp = -0.2;
uint16 new_timet;

/*
CY_ISR(interCounter) {
    uint16 captureVal;
    char strbuf[16];

    captureVal = Timer_ReadCapture();

    sprintf(strbuf, "Count: %u", captureVal);

    char strbuf2[32];
    sprintf(strbuf2,"%s\r\n",strbuf);
    UART_PutString(strbuf2);
```

```
    LCD_ClearDisplay();
    LCD_Position(0,0);
    LCD_PrintString(strbuf);

    PWM_Motor_WriteCompare((uint8_t) 20);
} */

/*
CY_ISR(control) {
    char strbuf[32];

    new_time = HE_Timer_ReadCapture();
    HE_Timer_ReadStatusRegister();

    if (old_time > new_time){
        dt = old_time - new_time;
    } else{
        dt = 65535 - new_time + old_time;
    }
    old_time = new_time;
    speed = 1300/(double)dt;
    error = target - speed;
    integration = integration + error;
    derivative = (speed - oldspeed)/(double)dt;
    if (speed != 0) {
        oldspeed = speed;
    }
    n = n0 + Kp*error + Ki*integration + Kd*derivative;
    if (n > 220) {
        n = 220;
    }
    if (n < 0) {
        n = 0;
    }
    PWM_Motor_WriteCompare((uint8_t) n);

    sprintf(strbuf, "Speed: %lf\r\n", speed);
    UART_PutString(strbuf);
    oldspeed = speed;
}
*/

CY_ISR(turn) {
    /* new_timet is the time from the edge of the line of video to the
left edge of the black line */
    new_timet = 65535 - Timer_Edge_ReadCapture();

    sprintf(strbuf2, "Count: %u \r\n", new_timet);
    UART_PutString(strbuf2);
```

```
    /* 487 is the count for the timer that represents straight */
    /* Only the error term was needed to complete PID control */
    PWM_Steer = 110 + kp * (487 - (double)new_timet);

    sprintf(strbuf2, "PWM: %f \r\n", PWM_Steer);
    UART_PutString(strbuf2);

    PWM_Servo_WriteCompare((uint8)PWM_Steer);
    Timer_Edge_ReadStatusRegister();
}

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    /* Place your initialization/startup code here (e.g.
MyInst_Start()) */
    UART_Start();
    LCD_Start();
    LCD_Position(0,0);
    LCD_PrintString("Hello World");
    HE_Timer_Start();
    PWM_Motor_Start();
    Timer_Edge_Start();
    PWM_Servo_Start();
    Comp_Black_Start();

    Ref_Start();
    //HE_Interrupt_Start();
    //HE_Interrupt_SetVector(control);
    turn_Start();
    turn_SetVector(turn);

    char strbuf2[32];
    sprintf(strbuf2,"test\r\n");
    UART_PutString(strbuf2);

    for(;;)
    {
        /* Place your application code here. */
    }
}

/* [] END OF FILE */
```

1.6.3 Arduino connections (Siren_LCD_Comm.c)

```
#include <SPI.h>
#include "LCD_Driver.h"
#include "GUI_Paint.h"
#include "image.h"

void setup()
{
  pinMode(3, OUTPUT); // Set digital pin 3 as output
  pinMode(5, OUTPUT); // Set digital pin 3 as output
  analogWrite(3, 230);
  analogWrite(5, 230);

  Serial.begin(115200);
  Serial3.begin(115200);

  Config_Init();
  LCD_Init();
  LCD_Clear(BLUE);
  Paint_NewImage(LCD_WIDTH, LCD_HEIGHT, 0, WHITE);
  Paint_Clear(WHITE);
  Paint_DrawString_EN(30, 10, "FIRE", &Font24, BLUE, RED);
}
void loop()
{
    // Siren generation
  static unsigned long lastToggle = 0;
  static bool is300Hz = true;
  float highestTemp = 0.0;
  bool siren = true;

  // Toggling frequency of the PWM signal every 500 milliseconds
  if (millis() - lastToggle >= 500){
    if (siren) {
      is300Hz = !is300Hz;
      analogWrite(3, is300Hz ? 60 : 230);
      analogWrite(5, is300Hz ? 60 : 230);
      lastToggle = millis();
    }
  }

  Paint_DrawString_EN(30, 10, "123", &Font24, BLUE, RED);

  if (Serial3.available() > 0) {
    // Send data only when you receive data
    uint8_t inChar = Serial3.read();
    if (inChar == 1) {
      siren = false;
      Serial.println("sirenEnd");
    }
  }
```

```
}


/
**********************************************************************
**********************************
  END FILE
**********************************************************************
**********************************/
```

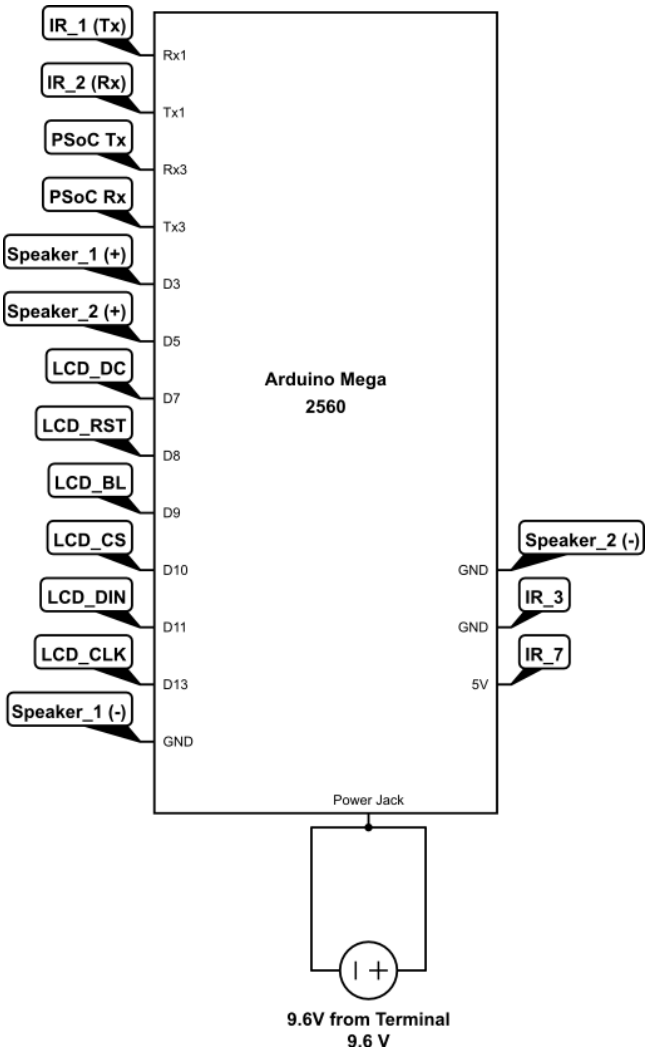


Figure 15: Arduino Pins

1.6.4 Arduino code for thermal image processing (T33_processing.c)

```
/*
 * This code allows you to check if the UART communication between the
TeraRanger and the Arduino Mega board works.
 * You just need to upload the code in the board with the TeraRanger
connected on pins TX1, RX1, GND and 5V. Make sure to connect the
signal RX of TeraRanger to TX1 of your Arduino and vice versa TX to
RX1).
 * If the supply of the TeraRanger is stopped or interrupted while the
code is running, just press the reset button on the board.
 * To check if the communication is working the best is to open the
serial monitor and check if the highest temperature is printed.
 */


// Create a Cyclic Redundancy Checks table used in the "crc32"
function
static const uint32_t crc32_table[] = {
    0x00000000, 0x04C11DB7, 0x09823B6E, 0x0D4326D9,
    0x130476DC, 0x17C56B6B, 0x1A864DB2, 0x1E475005,
    0x2608EDB8, 0x22C9F00F, 0x2F8AD6D6, 0x2B4BCB61,
    0x350C9B64, 0x31CD86D3, 0x3C8EA00A, 0x384FBDBD,
    0x4C11DB70, 0x48D0C6C7, 0x4593E01E, 0x4152FDA9,
    0x5F15ADAC, 0x5BD4B01B, 0x569796C2, 0x52568B75,
    0x6A1936C8, 0x6ED82B7F, 0x639B0DA6, 0x675A1011,
    0x791D4014, 0x7DDC5DA3, 0x709F7B7A, 0x745E66CD,
    0x9823B6E0, 0x9CE2AB57, 0x91A18D8E, 0x95609039,
    0x8B27C03C, 0x8FE6DD8B, 0x82A5FB52, 0x8664E6E5,
    0xBE2B5B58, 0xBAEA46EF, 0xB7A96036, 0xB3687D81,
    0xAD2F2D84, 0xA9EE3033, 0xA4AD16EA, 0xA06C0B5D,
    0xD4326D90, 0xD0F37027, 0xDDB056FE, 0xD9714B49,
    0xC7361B4C, 0xC3F706FB, 0xCEB42022, 0xCA753D95,
    0xF23A8028, 0xF6FB9D9F, 0xFBB8BB46, 0xFF79A6F1,
    0xE13EF6F4, 0xE5FFEB43, 0xE8BCCD9A, 0xEC7DD02D,
    0x34867077, 0x30476DC0, 0x3D044B19, 0x39C556AE,
    0x278206AB, 0x23431B1C, 0x2E003DC5, 0x2AC12072,
    0x128E9DCF, 0x164F8078, 0x1B0CA6A1, 0x1FCDBB16,
    0x018AEB13, 0x054BF6A4, 0x0808D07D, 0x0CC9CDCA,
    0x7897AB07, 0x7C56B6B0, 0x71159069, 0x75D48DDE,
    0x6B93DDDB, 0x6F52C06C, 0x6211E6B5, 0x66D0FB02,
    0x5E9F46BF, 0x5A5E5B08, 0x571D7DD1, 0x53DC6066,
    0x4D9B3063, 0x495A2DD4, 0x44190B0D, 0x40D816BA,
    0xACA5C697, 0xA864DB20, 0xA527FDF9, 0xA1E6E04E,
    0xBFA1B04B, 0xBB60ADFC, 0xB6238B25, 0xB2E29692,
    0x8AAD2B2F, 0x8E6C3698, 0x832F1041, 0x87EE0DF6,
    0x99A95DF3, 0x9D684044, 0x902B669D, 0x94EA7B2A,
    0xE0B41DE7, 0xE4750050, 0xE9362689, 0xEDF73B3E,
    0xF3B06B3B, 0xF771768C, 0xFA325055, 0xFEF34DE2,
    0xC6BCF05F, 0xC27DEDE8, 0xCF3ECB31, 0xCBFFD686,
    0xD5B88683, 0xD1799B34, 0xDC3ABDED, 0xD8FBA05A,
    0x690CE0EE, 0x6DCDFD59, 0x608EDB80, 0x644FC637,
```

```
    0x7A089632, 0x7EC98B85, 0x738AAD5C, 0x774BB0EB,
    0x4F040D56, 0x4BC510E1, 0x46863638, 0x42472B8F,
    0x5C007B8A, 0x58C1663D, 0x558240E4, 0x51435D53,
    0x251D3B9E, 0x21DC2629, 0x2C9F00F0, 0x285E1D47,
    0x36194D42, 0x32D850F5, 0x3F9B762C, 0x3B5A6B9B,
    0x0315D626, 0x07D4CB91, 0x0A97ED48, 0x0E56F0FF,
    0x1011A0FA, 0x14D0BD4D, 0x19939B94, 0x1D528623,
    0xF12F560E, 0xF5EE4BB9, 0xF8AD6D60, 0xFC6C70D7,
    0xE22B20D2, 0xE6EA3D65, 0xEBA91BBC, 0xEF68060B,
    0xD727BBB6, 0xD3E6A601, 0xDEA580D8, 0xDA649D6F,
    0xC423CD6A, 0xC0E2D0DD, 0xCDA1F604, 0xC960EBB3,
    0xBD3E8D7E, 0xB9FF90C9, 0xB4BCB610, 0xB07DABA7,
    0xAE3AFBA2, 0xAAFBE615, 0xA7B8C0CC, 0xA379DD7B,
    0x9B3660C6, 0x9FF77D71, 0x92B45BA8, 0x9675461F,
    0x8832161A, 0x8CF30BAD, 0x81B02D74, 0x857130C3,
    0x5D8A9099, 0x594B8D2E, 0x5408ABF7, 0x50C9B640,
    0x4E8EE645, 0x4A4FFBF2, 0x470CDD2B, 0x43CDC09C,
    0x7B827D21, 0x7F436096, 0x7200464F, 0x76C15BF8,
    0x68860BFD, 0x6C47164A, 0x61043093, 0x65C52D24,
    0x119B4BE9, 0x155A565E, 0x18197087, 0x1CD86D30,
    0x029F3D35, 0x065E2082, 0x0B1D065B, 0x0FDC1BEC,
    0x3793A651, 0x3352BBE6, 0x3E119D3F, 0x3AD08088,
    0x2497D08D, 0x2056CD3A, 0x2D15EBE3, 0x29D4F654,
    0xC5A92679, 0xC1683BCE, 0xCC2B1D17, 0xC8EA00A0,
    0xD6AD50A5, 0xD26C4D12, 0xDF2F6BCB, 0xDBEE767C,
    0xE3A1CBC1, 0xE760D676, 0xEA23F0AF, 0xEEE2ED18,
    0xF0A5BD1D, 0xF464A0AA, 0xF9278673, 0xFDE69BC4,
    0x89B8FD09, 0x8D79E0BE, 0x803AC667, 0x84FBDBD0,
    0x9ABC8BD5, 0x9E7D9662, 0x933EB0BB, 0x97FFAD0C,
    0xAFB010B1, 0xAB710D06, 0xA6322BDF, 0xA2F33668,
    0xBCB4666D, 0xB8757BDA, 0xB5365D03, 0xB1F740B4
};

/*
 * Brief : Calculate a Cyclic Redundancy Checks of 32 bits
 * Param1 : (*p) pointer to data buffer
 * Param2 : (len) number of bytes in the data buffer
 * Return : (crc) checksum calculated locally
 */
uint32_t crc32(uint8_t *p, size_t len) {
    uint32_t crc = 0xFFFFFFFF;
    while (len--) {
        crc = crc32_table[(crc ^ *p++) & 0xFF] ^ (crc >> 8);
    }
    return ~crc; // Final XOR
}

// Initialize variables
const int BUFFER_LENGTH = 2070;
uint8_t Framereceived[BUFFER_LENGTH];// The variable "Framereceived[]"
```

1.6.5 The Fire truck Project

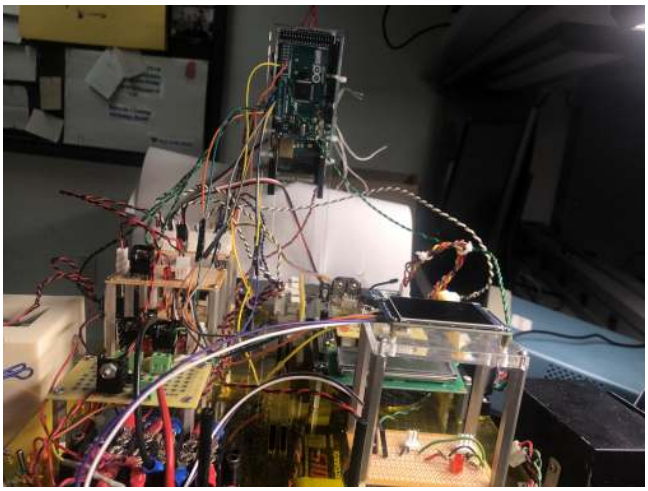


Figure 16: A view from the back

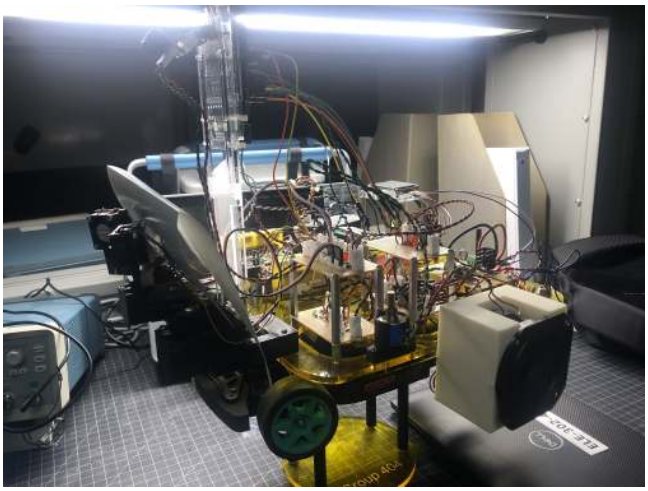


Figure 17: A view from a diagonal perspective

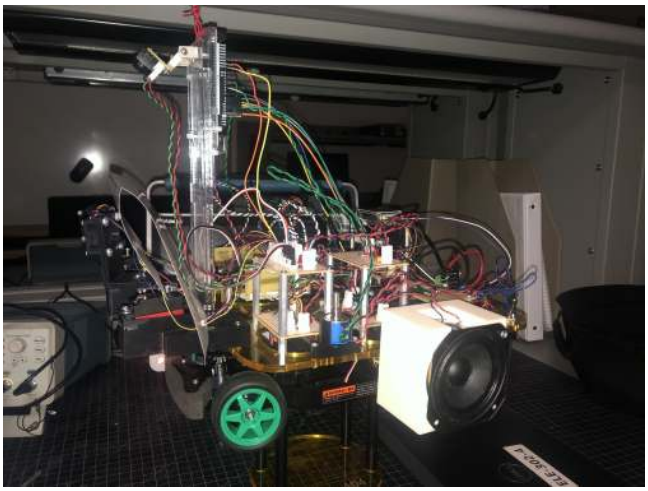


Figure 18: A view from a side perspective

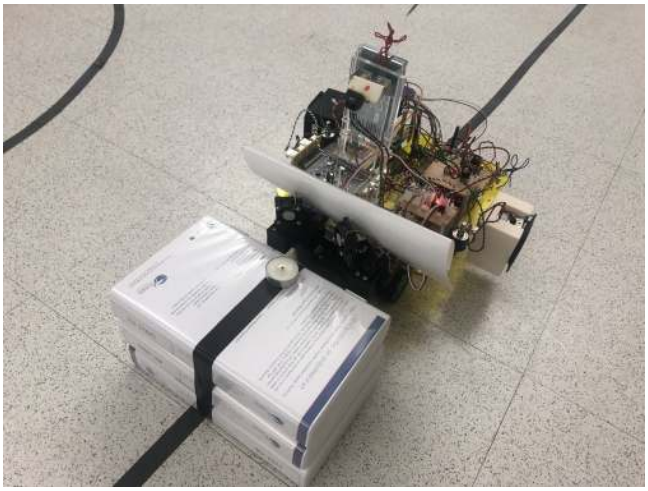


Figure 19: The position at the end of a trial

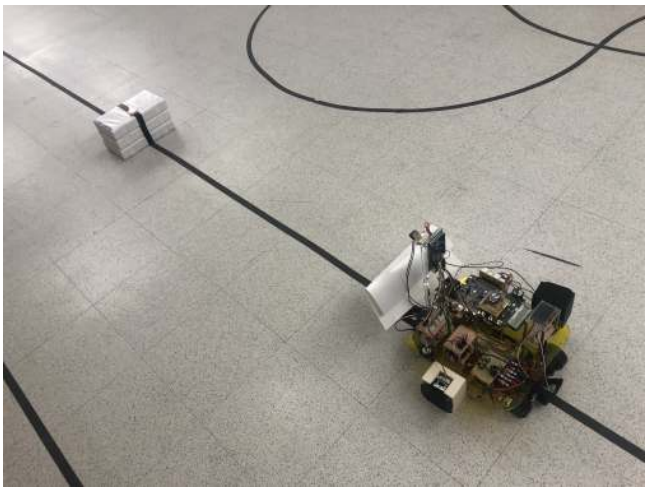


Figure 20: The position at the start of a trial