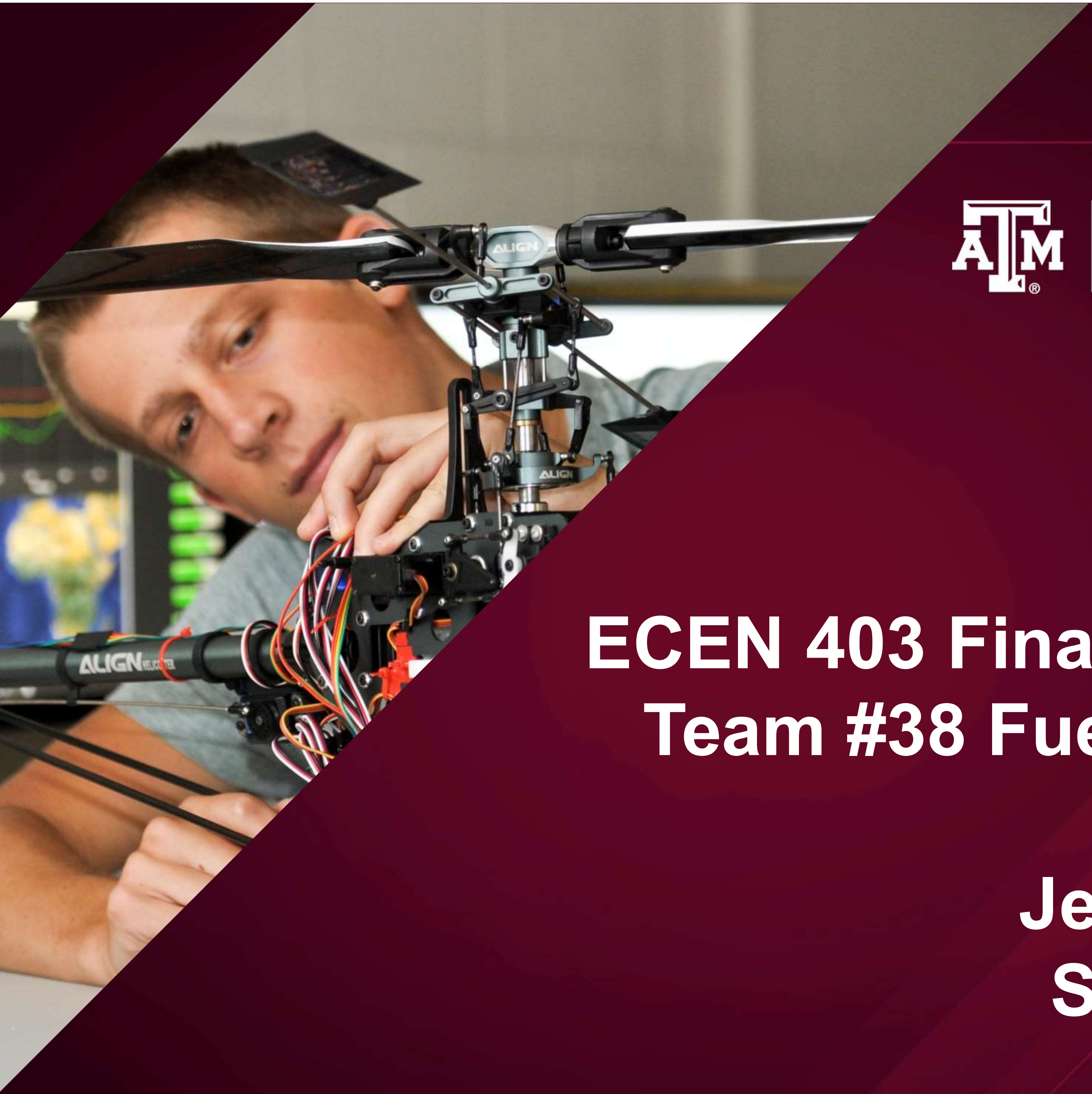


Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

ECEN 403 Final Presentation
Team #38 Fuel Cell Monitor

Rana Kortam
Jessica Odutola
Sameer Osama
Russell Wells
John Lusher

Project description

- Problem statement: A single fuel cell is an easy power source to monitor, but to achieve any level of real usable power they must be connected in a stack. The goal of this project is to design a monitor that displays individual cell voltages within a stack and warns the user of cell abnormalities as well as which cell requires maintenance or attention.

Diagram of subsystems and interface

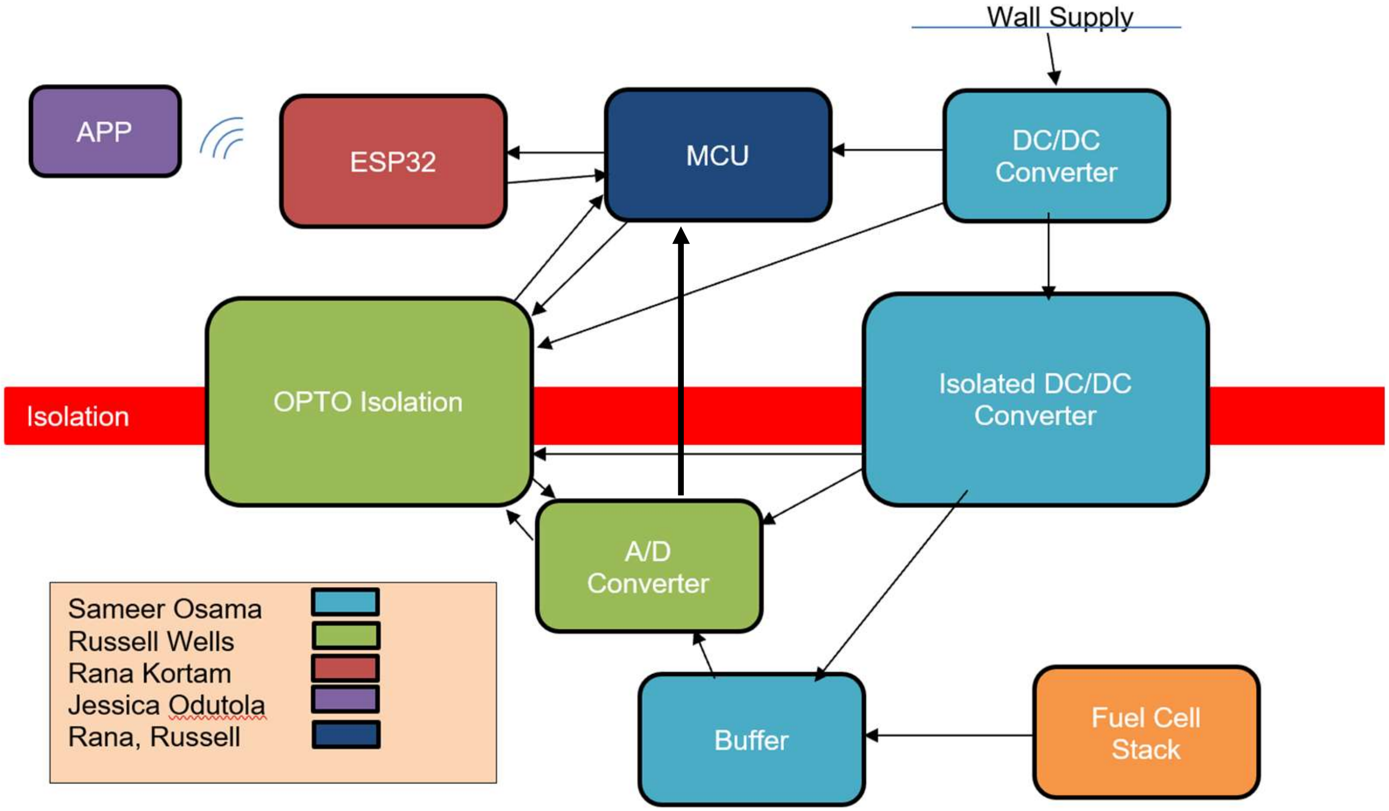
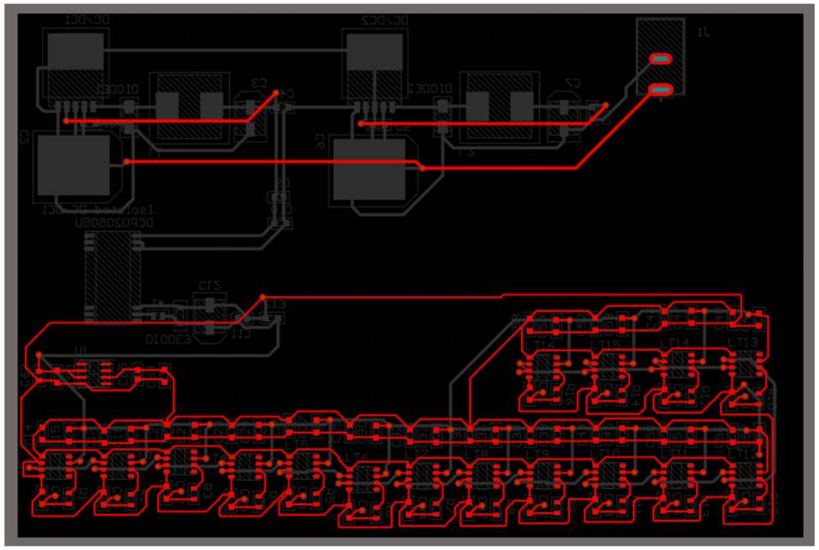


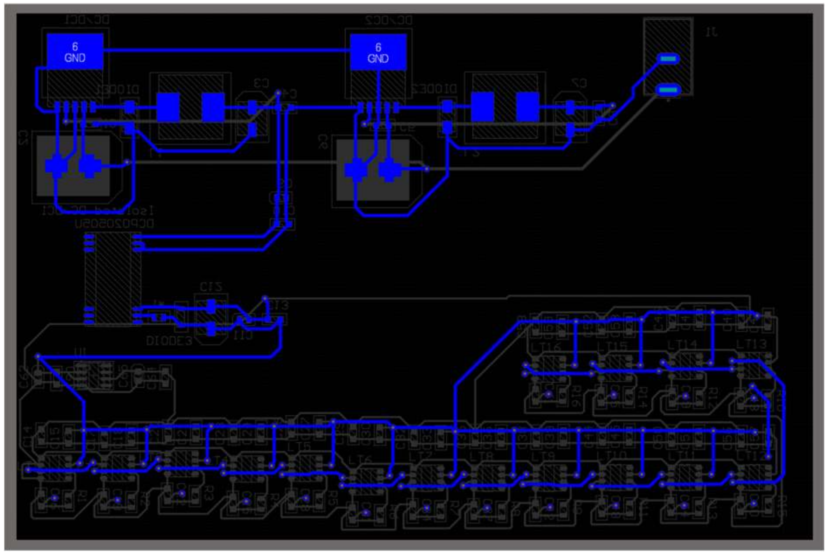
Figure 2. Block Diagram of System

403 Deliverables (Power Supply)

- Design PCB in Altium
- Order PCB and solder components
- Test output voltages



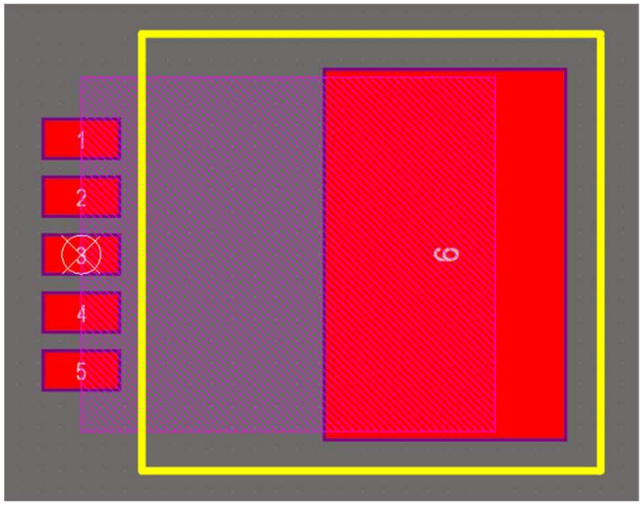
Top Layer

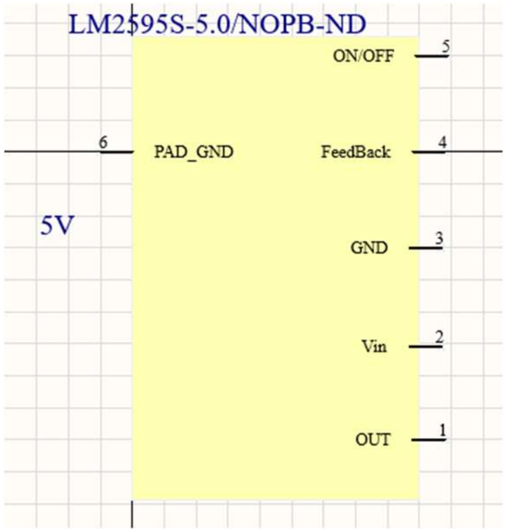


Bottom Layer

Challenges (Power Supply)

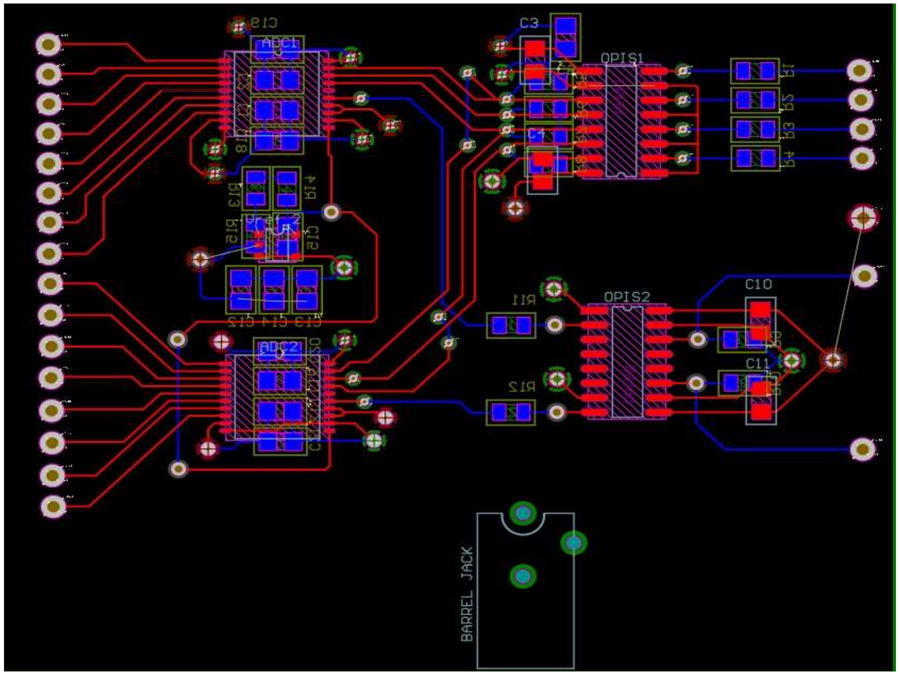
- Learning how to make schematics and footprints
- Using a new software (Altium)
- Figuring out what specific parts to order





Internal Data Transfer and Manipulation Deliverables

- Create PCB Design.
- Create Breadboard Test Circuit
- Confirm Isolator Circuit's Ability To Pass Digital Information.
Confirm Isolator Circuit's Ability to Control ADC.



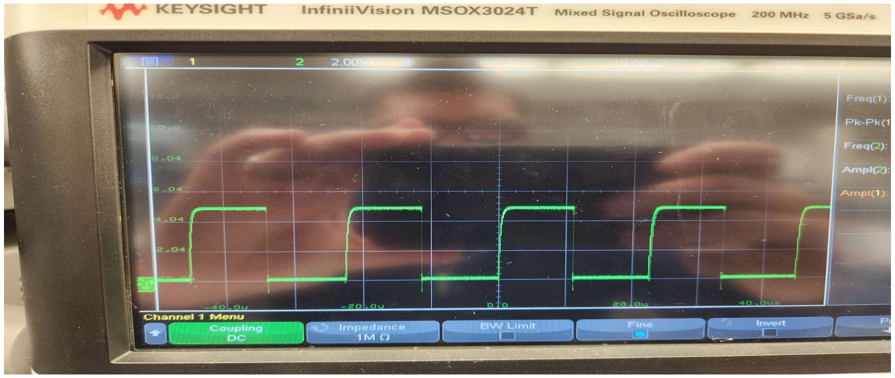
Internal Data Transfer and Manipulation

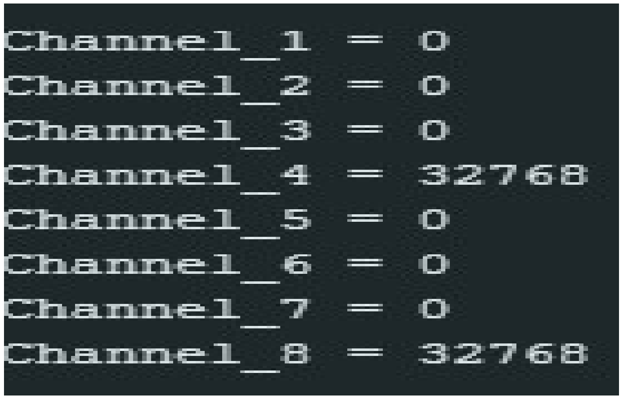
CHALLENGES

1. Isolator Distorts Digital Signal
2. Switching Speed of isolator is much lower than anticipated
3. Learning To code with Arduino and use SPI Library.

SOLUTIONS

1. Reduce Clock Frequency from 2MHz to 125kHz (Ongoing)
2. Same solution as 1 but the frequency is slightly higher than 1/24th the capability of the ADC.
3. Continuing to learn from Arduino website and forums.





```
Channel_1 = 0
Channel_2 = 0
Channel_3 = 0
Channel_4 = 32768
Channel_5 = 0
Channel_6 = 0
Channel_7 = 0
Channel_8 = 32768
```

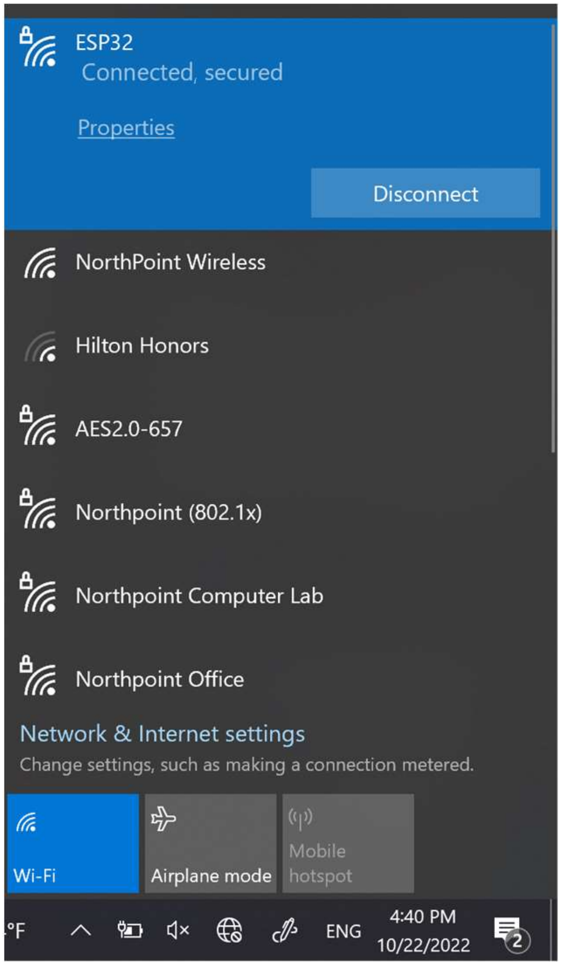
ESP32

403 Deliverables

- ESP32 to connect to WiFi
- UART communication for ESP32
- Connect ESP32 to database

Challenges

- Learning the ESP-IDE
- Connection to database



ESP32
Connected, secured

NorthPoint Wireless

Hilton Honors

AES2.0-657

Northpoint (802.1x)

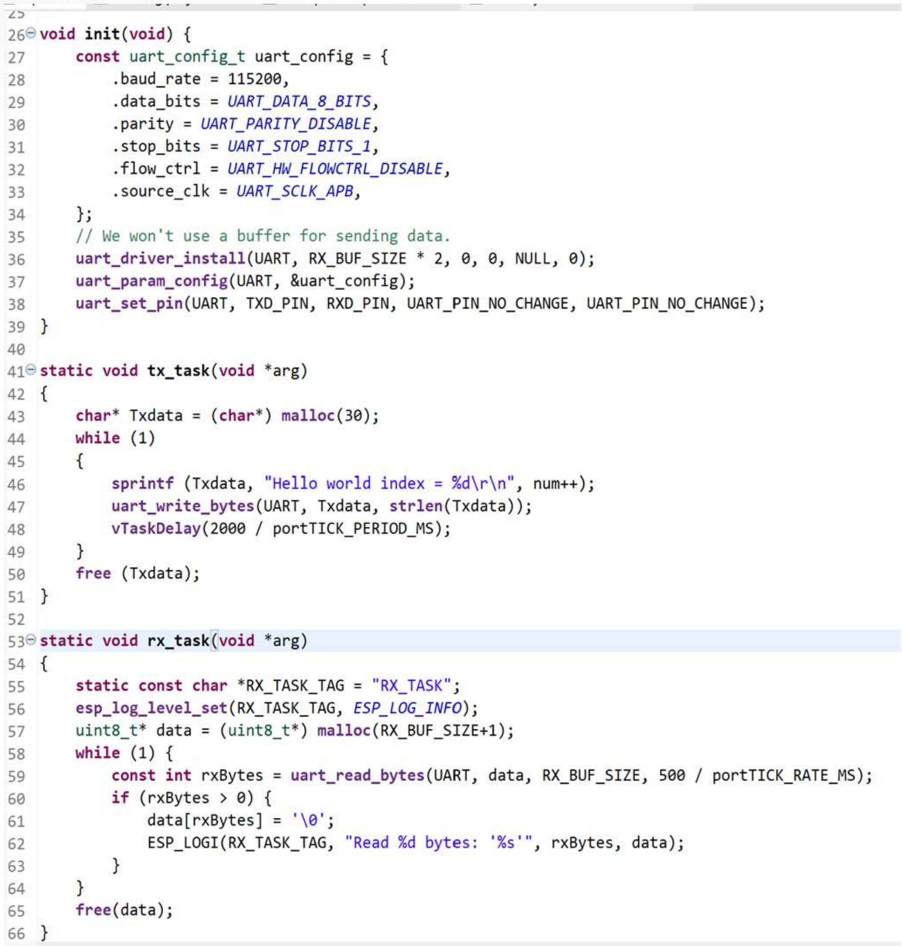
Northpoint Computer Lab

Northpoint Office

Network & Internet settings
Change settings, such as making a connection metered.

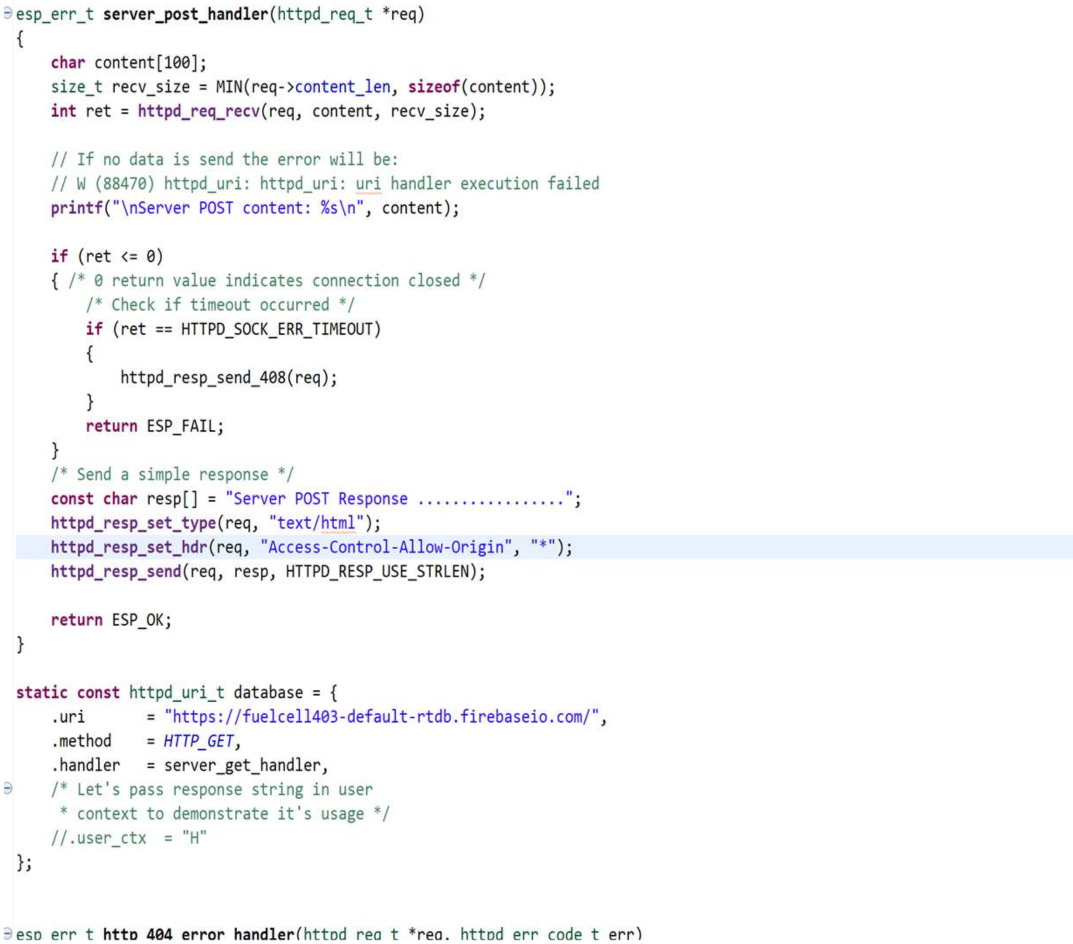
ESP32

UART Code



```c
void init(void) {
    const uart_config_t uart_config = {
        .baud_rate = 115200,
        .data_bits = UART_DATA_8_BITS,
        .parity = UART_PARITY_DISABLE,
        .stop_bits = UART_STOP_BITS_1,
        .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
        .source_clk = UART_SCLK_APB,
    };
    // We won't use a buffer for sending data.
    uart_driver_install(UART, RX_BUF_SIZE * 2, 0, 0, NULL, 0);
    uart_param_config(UART, &uart_config);
    uart_set_pin(UART, TXD_PIN, RXD_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
}

static void tx_task(void *arg)
{
    char* Txdata = (char*) malloc(30);
    while (1)
    {
        sprintf (Txdata, "Hello world index = %d\r\n", num++);
        uart_write_bytes(UART, Txdata, strlen(Txdata));
        vTaskDelay(2000 / portTICK_PERIOD_MS);
    }
    free (Txdata);
}

static void rx_task(void *arg)
{
    static const char *RX_TASK_TAG = "RX_TASK";
    esp_log_level_set(RX_TASK_TAG, ESP_LOG_INFO);
    uint8_t* data = (uint8_t*) malloc(RX_BUF_SIZE+1);
    while (1) {
        const int rxBytes = uart_read_bytes(UART, data, RX_BUF_SIZE, 500 / portTICK_RATE_MS);
        if (rxBytes > 0) {
            data[rxBytes] = '\0';
            ESP_LOGI(RX_TASK_TAG, "Read %d bytes: '%s'", rxBytes, data);
        }
    }
    free(data);
}
```

Database connection code



```c
esp_err_t server_post_handler(httpd_req_t *req)
{
    char content[100];
    size_t recv_size = MIN(req->content_len, sizeof(content));
    int ret = httpd_req_recv(req, content, recv_size);

    // If no data is send the error will be:
    // W (88470) httpd_uri: httpd_uri: uri handler execution failed
    printf("\nServer POST content: %s\n", content);

    if (ret <= 0)
    { /* 0 return value indicates connection closed */
        /* Check if timeout occurred */
        if (ret == HTTPD_SOCK_ERR_TIMEOUT)
        {
            httpd_resp_send_408(req);
        }
        return ESP_FAIL;
    }
    /* Send a simple response */
    const char resp[] = "Server POST Response ................";
    httpd_resp_set_type(req, "text/html");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    httpd_resp_send(req, resp, HTTPD_RESP_USE_STRLEN);

    return ESP_OK;
}

static const httpd_uri_t database = {
    .uri       = "https://fuelcell403-default-rtdb.firebaseio.com/",
    .method    = HTTP_GET,
    .handler   = server_get_handler,
    /* Let's pass response string in user
     * context to demonstrate it's usage */
    //.user_ctx  = "H"
};


esp_err_t http_404_error_handler(httpd_req_t *req, httpd_err_code_t err)
```

PIC32

403 Deliverables

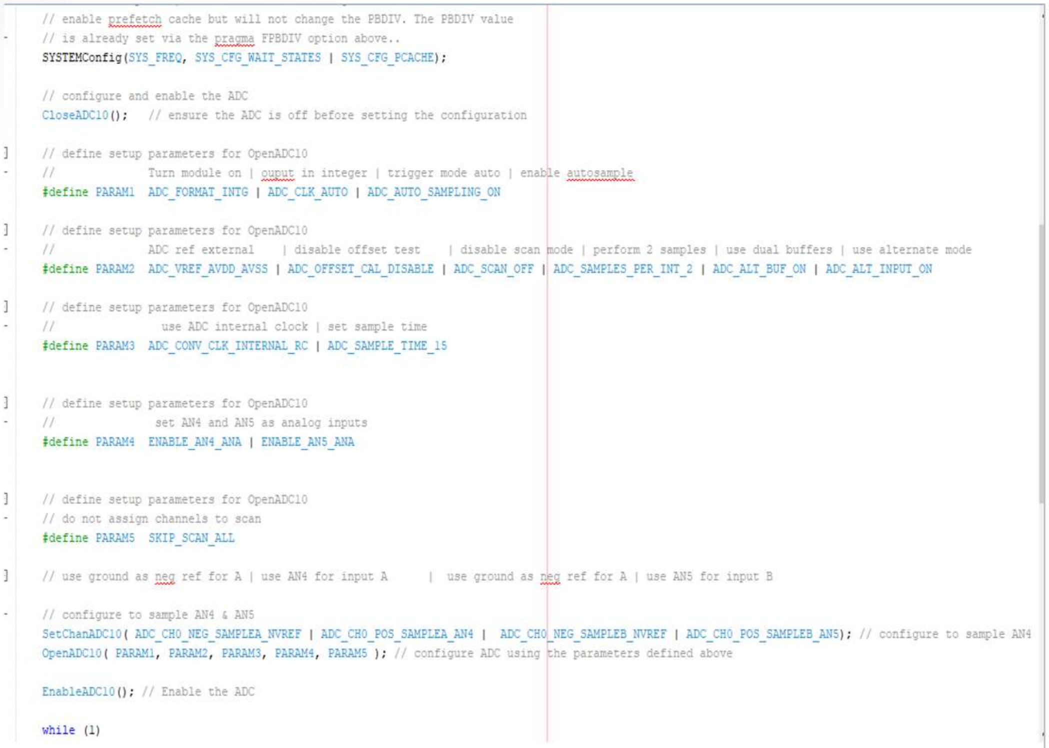
- UART communication
- A/D converter code
- Array of fuel-cell voltages

Challenges

- Learning MPLAB IDE
- Fried the PIC32 chip

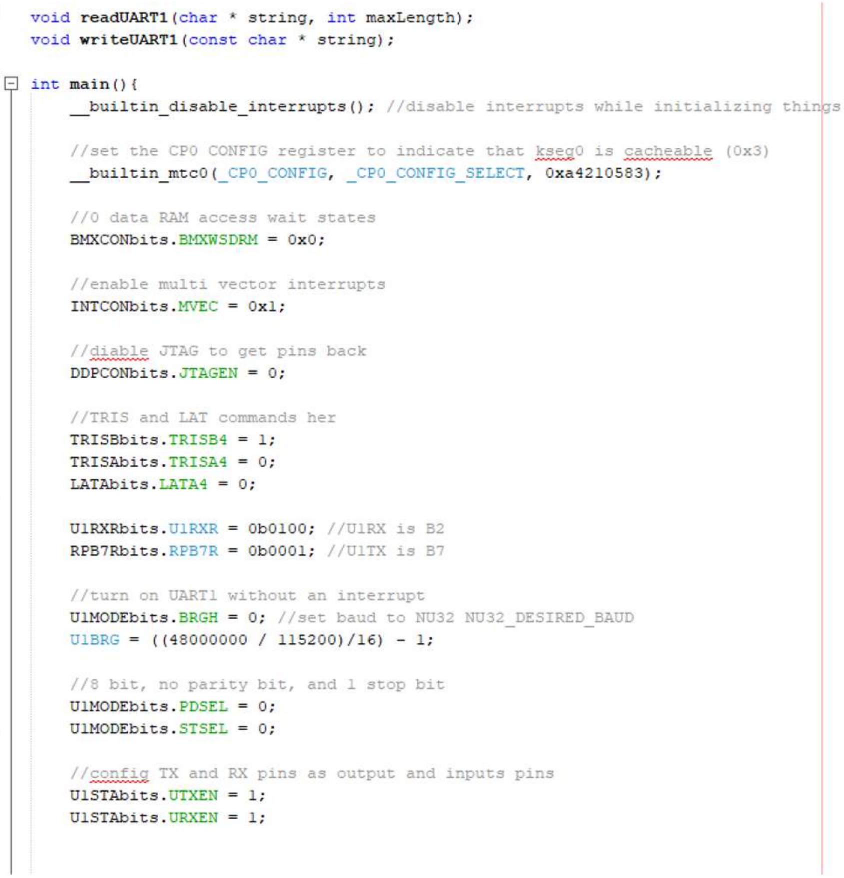
PIC32

AD Converter Code



```
// enable prefetch cache but will not change the PBDIV. The PBDIV value
// is already set via the pragma FPBDIV option above..
SYSTEMConfig(SYS_FREQ, SYS_CFG_WAIT_STATES | SYS_CFG_PCACHE);

// configure and enable the ADC
CloseADC10();   // ensure the ADC is off before setting the configuration

// define setup parameters for OpenADC10
//              Turn module on | ouput in integer | trigger mode auto | enable autosample
#define PARAM1  ADC_FORMAT_INTG | ADC_CLK_AUTO | ADC_AUTO_SAMPLING_ON

// define setup parameters for OpenADC10
//              ADC ref external    | disable offset test    | disable scan mode | perform 2 samples | use dual buffers | use alternate mode
#define PARAM2  ADC_VREF_AVDD_AVSS | ADC_OFFSET_CAL_DISABLE | ADC_SCAN_OFF | ADC_SAMPLES_PER_INT_2 | ADC_ALT_BUF_ON | ADC_ALT_INPUT_ON

// define setup parameters for OpenADC10
//                use ADC internal clock | set sample time
#define PARAM3  ADC_CONV_CLK_INTERNAL_RC | ADC_SAMPLE_TIME_15


// define setup parameters for OpenADC10
//               set AN4 and AN5 as analog inputs
#define PARAM4  ENABLE_AN4_ANA | ENABLE_AN5_ANA


// define setup parameters for OpenADC10
// do not assign channels to scan
#define PARAM5  SKIP_SCAN_ALL

// use ground as neg ref for A | use AN4 for input A      |  use ground as neg ref for A | use AN5 for input B

// configure to sample AN4 & AN5
SetChanADC10( ADC_CH0_NEG_SAMPLEA_NVREF | ADC_CH0_POS_SAMPLEA_AN4 |  ADC_CH0_NEG_SAMPLEB_NVREF | ADC_CH0_POS_SAMPLEB_AN5); // configure to sample AN4
OpenADC10( PARAM1, PARAM2, PARAM3, PARAM4, PARAM5 ); // configure ADC using the parameters defined above

EnableADC10(); // Enable the ADC

while (1)
```

UART Code



```
void readUART1(char * string, int maxLength);
void writeUART1(const char * string);

int main(){
    __builtin_disable_interrupts(); //disable interrupts while initializing things

    //set the CP0 CONFIG register to indicate that kseg0 is cacheable (0x3)
    __builtin_mtc0(_CP0_CONFIG, _CP0_CONFIG_SELECT, 0xa4210583);

    //0 data RAM access wait states
    BMXCONbits.BMXWSDRM = 0x0;

    //enable multi vector interrupts
    INTCONbits.MVEC = 0x1;

    //diable JTAG to get pins back
    DDPCONbits.JTAGEN = 0;

    //TRIS and LAT commands her
    TRISBbits.TRISB4 = 1;
    TRISAbits.TRISA4 = 0;
    LATAbits.LATA4 = 0;

    U1RXRbits.U1RXR = 0b0100; //U1RX is B2
    RPB7Rbits.RPB7R = 0b0001; //U1TX is B7

    //turn on UART1 without an interrupt
    U1MODEbits.BRGH = 0; //set baud to NU32 NU32_DESIRED_BAUD
    U1BRG = ((48000000 / 115200)/16) - 1;

    //8 bit, no parity bit, and 1 stop bit
    U1MODEbits.PDSEL = 0;
    U1MODEbits.STSEL = 0;

    //config TX and RX pins as output and inputs pins
    U1STAbits.UTXEN = 1;
    U1STAbits.URXEN = 1;
```

Android Application

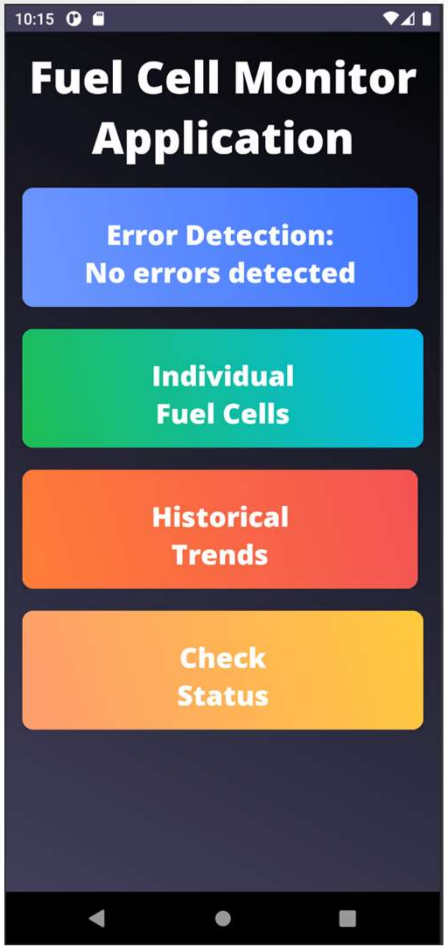
403 Deliverables

- Complete Layout of Android Application
- Database Storage and Connection to App
- Alert functionality for user

Challenges

- Database Connection

Android Application



10:15

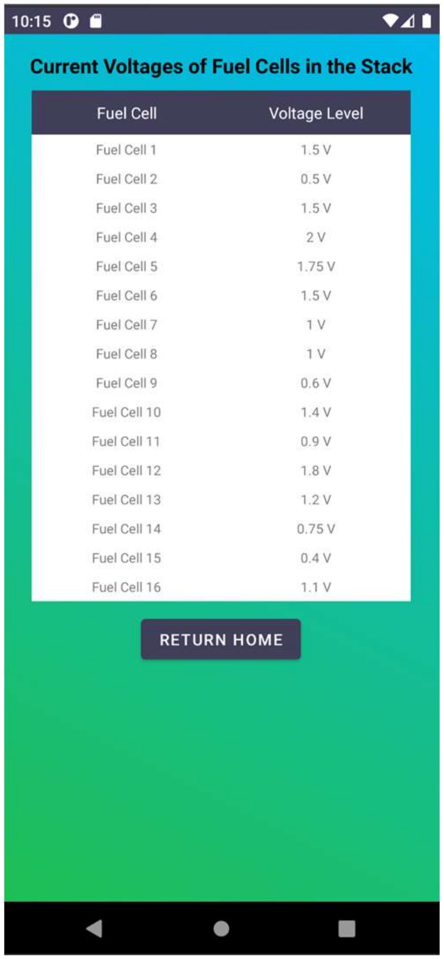
Fuel Cell Monitor Application

Error Detection:
No errors detected

Individual
Fuel Cells

Historical
Trends

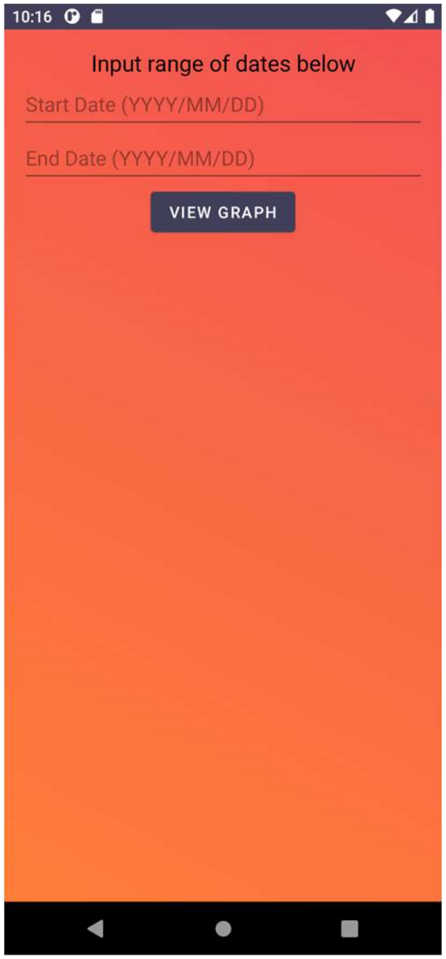
Check
Status



10:15

Current Voltages of Fuel Cells in the Stack

Fuel Cell	Voltage Level
Fuel Cell 1	1.5 V
Fuel Cell 2	0.5 V
Fuel Cell 3	1.5 V
Fuel Cell 4	2 V
Fuel Cell 5	1.75 V
Fuel Cell 6	1.5 V
Fuel Cell 7	1 V
Fuel Cell 8	1 V
Fuel Cell 9	0.6 V
Fuel Cell 10	1.4 V
Fuel Cell 11	0.9 V
Fuel Cell 12	1.8 V
Fuel Cell 13	1.2 V
Fuel Cell 14	0.75 V
Fuel Cell 15	0.4 V
Fuel Cell 16	1.1 V

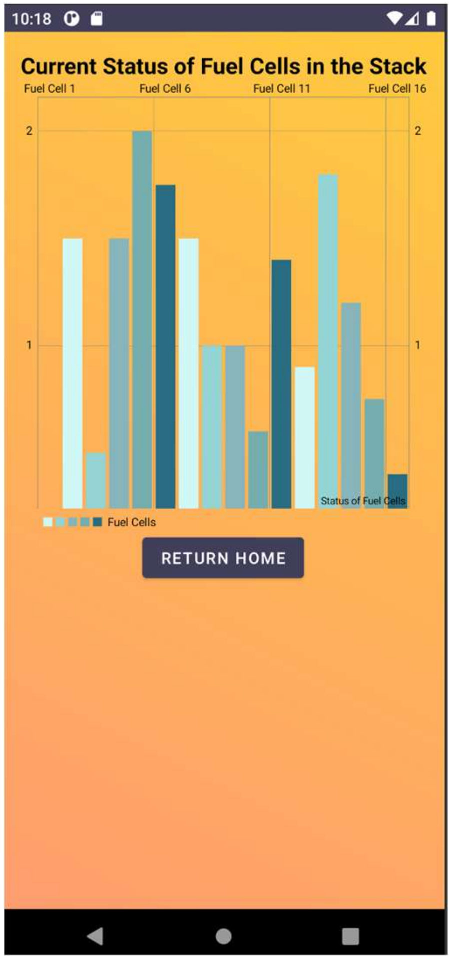


10:16

Input range of dates below

Start Date (YYYY/MM/DD)

End Date (YYYY/MM/DD)



10:18

Current Status of Fuel Cells in the Stack

Execution and Validation Status

	9/5/2022	9/12/2022	9/19/2022	9/26/2022	10/3/2022	10/10/2022	10/17/2022	10/24/2022	10/31/2022	11/7/2022	11/14/2022	11/21/2022	11/28/2022	DATE
TEAM DELIVERABLES														
Understand Project Problem	🟩	🟩												
Project design Overview	🟩	🟩											🟩	Completed
Divide Into Subsystems	🟩	🟩											🟨	In Progress
ConOps Report			🟩	🟩									⬛	Not Started
Create Major Parts List			🟩	🟩	🟩	🟩							🟥	Behind Schedule
FSR, ICD Report			🟩	🟩	🟩	🟩								
Midterm Presentation			🟩	🟩	🟩	🟩								
Order Major Parts					🟩	🟩								
Status Update Presentation					🟨	🟨	🟨	🟨						
Final Presentation											🟨	🟨	🟨	
Final Demo											🟨	🟨	🟨	
Final Report											🟨	🟨	🟨	
POWER SUBSYSTEM														
Determine IC Components	🟩	🟩												
Design Schematics		🟩	🟩											
Order IC components				🟩	🟩									
Create PCB footprints in Altium						🟨	🟨	🟨	🟨					
Create PCB design in Altium									🟩	🟩				
Make Gerber files and send to FEDC										🟩	🟩			
Test components on circuit board											🟨	🟨	🟨	
INTERNAL SIGNAL SUBSYSTEM														
Determine IC Components		🟩	🟩											
Design System				🟩	🟩									
Order Components						🟩	🟩							
Create PCB Schematic						🟩	🟩	🟩	🟩	🟩				
Assemble and Test Demo								🟨	🟨	🟨	🟨	🟨	🟨	
Create PCB Design									🟥	🟥				
Order PCB													🟨	
MICRO CONTROLLER SUBSYSTEM														
Determine Microcontrollers in use	🟩	🟩	🟩	🟩										
Learn IDE to code microcontroller					🟩	🟩								
Implement "Hello World" on ESP32							🟩							
WiFi connection on ESP32							🟩	🟩						
UART on ESP32							🟩	🟩						
UART on PIC32								🟩	🟩					
Array code for PIC32									🟩	🟩				
AD Converter code											🟩	🟩		
Connect ESP32 to database											🟨	🟨	🟨	
APP SUBSYSTEM														
App Displays "Hello World"				🟩	🟩									
App Displays Home Page						🟨	🟨							
App Displays all pages needed								🟩						
AWS Database Created								🟩						
Tables Populated in Database									🟩					
Connect Database to App											🟨	🟨		
App Sends Alerts to Users										🟩				
App Works with Test Data											🟩	🟩		
WiFi Connection with Microcontroller											🟨	🟨	🟨	

Execution and Validation Status

Paragraph #	Test Name	Success Criteria	Methodology	Status	Responsible Engineer(s)
3.2.4.2	Power Devices On PCB	PCB transfers power without overheating or burnout	Power Board and watch, smell, listen	Untested	Russell, Sameer
3.2.1.1	Internal signal voltage range	System can properly handle the specified voltages with minimal difference between tests.	Introduce voltages of 0-4V and measure output signals	FAIL	Russell
3.2.1.1	Differential voltage tests	Pass a differential voltage through the Opamp buffer and receive the proper digital signal from the optoisolator	Introduce a range of voltages including edge cases and ensure proper output	Untested	Russell, Sameer
3.2.4.4	Android application graphical functionality	Application can properly display accurate voltage levels to user.	Use application on android device and verify volatages are accurately displayed	Tested	Jessica
3.2.4.4	Android Application alarm functionality	Application send alarm to user when voltage goes above or below ranges	Add set points to app and introduce alarm level voltages	Untested	Jessica
3.2.4.2	Power system functionality test	Power is applied from wall outlet and proper power transfer is read at outputs	Apply power to system and read voltage output at device trace	Untested	Sameer
3.2.4.1	Opamp system functionality test	Differential voltages are passed to the opamp and expected voltage is seen on the output	Power opamps and apply varrying differential voltages and read output voltage	Untested	Sameer
N/A	PIC32 Microcontroller functionality test	The code for recieving the voltage signal for data acquisition	PCB board and coding on IDE	In progress	Rana
N/A	ESP32 Microcontroller functionality test	The code for communicating with the application	PCB board and coding on IDE	Tested	Rana

Remaining Tasks

- Complete testing of internal signal system
- Complete Microcontroller/processor PCB design
- Integrate All subsystems onto a single PCB.
- Final Testing and Validation of Completed Internal System.
- Purchase system enclosure and assemble.