2/16/2025

- Met up as a group on a discord call from 3-5 PM. Talked about design choices, design review and how we are going to split work when presenting.
- Design modifications if proposal does not get accepted.
    - If more complexity is needed:
        - Add one more temperature sensor and one more moisture sensor.
            - We would then take the average between the two temperature sensors to get a more reliable reading and we could also do the same for the moisture sensor.
        - Design a UI app
            - This app would allow the user to turn "on" our system. Which means the system would start sensing the environment and heat up when necessary.
            - It would also allow the user to turn it off
            - The app can also show the user how much money / resources they would have saved compared to if they kept the heater running the whole time.
- We also went over the placement and wiring of the sensors
    - Option 1 -> Place moisture sensor in the middle of the plate and the temperature sensor under the plate, there would be something to insulate the temperature sensor so it does not get incorrect reading if the heat is on.
        - We could ask machine shop to poke a hole between the plate to get our wires through
    - Option 2 -> Use ESP32 bluetooth and wifi features to send data to the MCU, this would require a separate battery for each sensors since they will be separated.
- Before our demo, I am planning on researching more about the ESP32, and our main event loop and how it will calculate if we would need to turn the heater on.
    - How we decide to turn on the heat
        - If temperature < 0 Celsius AND moisture level is greater than the threshold
        - If temperature >= 0 Celsius OR moisture level is less than or equal to the threshold
    - Some pseudo code:

```
if (temperature < 0°C AND moisture_level > threshold):
    // Activate heating if freezing and moisture present
    if (heating_status == OFF):
        turn_on_heating()
        heating_status = ON
        log_event("Heating activated due to freezing and moisture detected.")
else if (temperature >= 0°C OR moisture_level <= threshold):
    // Deactivate heating if conditions are safe
    if (heating_status == ON):
        turn_off_heating()
        heating_status = OFF
        log_event("Heating deactivated - safe conditions.")
else:
    // Default case: system is in a neutral state, no action needed
    log_event("System in neutral state - no heating needed.")
```

- ESP32
    - Temperature Sensor
    - Moisture Sensor
    - MOSFET-Based Switching

2/17/2025 - Design Review

- Met up again before our demo to prepare, talked about placement of moisture sensors and temperature sensors as well. Moisture sensor can be difficult since we would need to detect water on the whole entire plate.
    - Moisture sensor placements
        - Drainage system to detect water?
        - Flat Moisture sensors that can detect water.
- During Demo / Post Demo
    - Advice that we were told:
        - We will lose a lot of heat heating the surface because of air.
        - Look into how heated driveways work to see if we can derive it
        - Look into how bathroom heated floors work to see if we can also derive it as well
        - Fix block diagrams, have a legend for different communication protocols. Have actual square boxes with its subsystem in smaller boxes
        - Fail safe option if there is too much ice and our system cannot catch up.
        - Look into the temperatures allowed for our peripherals and parts.
        - Placements of PCB, how much we will heat and how to scale it

2/19/2025

- Peer Reviewed GymHive Tracker group
- Different parts to consider:
    - Water sensor: https://www.amazon.com/dp/B0BXKMLB4D?&linkCode=sl1&tag=zlufy-20&linkId=c3cc423f6887383dbbef16c3d21264c7&language=en_US&ref_=as_li_ss_tl
    - Rain Sensor: https://www.amazon.com/dp/B0CM6224T2?linkCode=sl1&tag=zlufy-20&linkId=d6c3b90b94272148cb801cf57015b64e&language=en_US&ref_=as_li_ss_tl&th=1
- Goal for the end of this week is to figure out the exact parts to buy so we can start on our PCB design.
- Plan is to meet up on Friday to discuss with machine shop with how to approach the metal plate.
-

2/23/2025

- Meeting with group to discuss what parts are needed
- Our goal by the end of this meeting is to decide the parts we need so we can start creating the schematic for our PCB
- Goal on Thursday is to get our PCB Ready
- We need to have these parts here:

- User Interface

    - Button
    - Sensor LEDs
- Display? - difficult to have when demo or realistic situation
- Sensing Subsystem
    - Temperature Sensor x2
    - Humidity Sensor
- Control Subsystem
    - ESP32?
    - Mosfet switch
- Power Subsystem
    - Voltage Regulator
    - AC/DC Converter
- Will meet with machine shop guy tomorrow (Monday) 2-3 pm we will be free
    - Where to place temperature sensors
        - I think we have this figured out
    - Where to place moisture sensor
        - Conversation to ask to machine shop guy
    - What will our plate look like
    - Maybe what it could look like

2/24/2025

- In charge of UI and Control SubSystem
    - MCU
        - ESP32-WROOM-32 has wireless capabilties just for future implementation
    - Mosfet Switch
        - IRLZ44NPBF
            - Vds = 55V
                - **VDS_{DS}DS** stands for **Drain-to-Source Voltage**. It is the maximum voltage that can be applied between the **drain (D)** and **source (S)** terminals of the MOSFET **before breakdown occurs**.
            - Id (at Vgs=-10V) = 47 A
                - **What it means**: The maximum current the MOSFET can safely carry from **drain to source**.
                - **Why it matters**: If your circuit requires **20A**, you need a MOSFET with **ID_DD > 20A** to avoid overheating.
    - Questions
        - We need to figure out how much current / power is needed to heat up the plate before the PCB so we know which peripherals to pick in a power subsystem.
            - Size of the plate
        - How much heat will our MOSFET generate and will we need heat sinks for that?
            - Heatsinks?
        - Moisture sensor placement
        - Stainless steel or Aluminimum
        - What heat cartridge to use
        - How realistic should this be?

    - Brainstorm
        - Railing on the bridge to place temperature sensor
        - Temperature Sensor at the bottom above the heat cartridges

Notes from Talk with Gregg in Machine Shop

Types of metals available in machine shop:

- 60/61 aluminum
- Stainless steel

Dimensions:

- Range of thickness for stainless steel - 1 and a half to 3 mm. 1/16 of an inch and or 1/8th
- 2 to 1 width to length. 12 inch long 6 inch width
    - Heat cartridges and how much heat
    - 10 inch length, 5 inch width, 1/8 inch
    - Metal type - steel
    - Thermal paste
- We can use this volume and type of material to get the mass of the metal

Thermocouple - measures what it touches

2/25/25 to 2/26/25

Notes / Thoughts while drawing schematic for the ESP32

- ESP32-WROOM-32
    - Datasheet: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
    - Programming: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html
    - Vdd range: Min: 3.0V, Typical: 3.3V, Max: 3.6V
    - Operating Temperature -40 C to 85 C
- Power will be 3.3 V
- Which pins are necessary to program the ESP32?
- Maybe for breadboard demo we can get the breakout board to test our software to show our UI?
    - Ordered this for prototype and testing: https://www.amazon.com/ESP-WROOM-32-Development-Microcontroller-Integrated-Compatible/dp/B08D5ZD528/ref=sr_1_3?crid=2R6NFSPNZ4UIJ&dib=eyJ2IjoiMSJ9.qMJJKscaTbDZH8KOrPXaSnOgh0Rwn0wGCCSUTqJssNJWvhgioFRHfCsyDoVodLIm1otAgJ2cz5g56oUcIK6S2BM0lxS7vSMj_1RGHVS94wZapor5YIFnpghILK189X77DwEid3tgVJCXqLxZK103j0eb6plEZCZ7ilOwI90qEdXmE3JMJT68Yu-DIUEuAJjWcGY61fKjhaYkZJ7T7GEw0D-sw4_fDt4bSxIifrXKnSk.ETzFxQSIHW3uG14JOZa1bLh6CFPtJtWteeAkEKG8Vok&dib_tag=se&keywords=ESP32-WROOM-32&qid=1740541125&sprefix=esp32-wroom-32%2Caps%2C120&sr=8-3&th=1
-

- Pins needed for 2x Temperature Sensor:
    - Part Number DS18B20
        - Datasheet: https://www.analog.com/media/en/technical-documentation/data-sheets/DS18B20.pdf
        - Water Proof
        - 1 Pin Interface
        - Supply Voltage
            - 3.0 Min
            - 5.5 Max V
    - PIN for Temp1 -> IO17
    - PIN for Temp2 -> IO19
- Pins needed for 1x Mouisture Sensor
    - Using the Simple Rain Sensor
        - 3.3V power
        - Analog pin
        - Digital Pin
        - GND
    - https://soldered.com/product/simple-rain-sensor/?srsltid=AfmBOop1M8PKcR7p4fGTHjeK9IQrmTysczny2TCHp53vRtrOn9PUprxF
    - Will use ADC Pin on ESP32 since we can convert it to digital
- Voltage Regulator
- Still need to figure out which pins are necessary for flashing / programming
    - Good reddit page flashing / programming the MCU
        - https://www.reddit.com/r/esp32/comments/ovkp96/how_to_flash_raw_esp32_board/
    - Good reddit page for example schematic
        - https://www.reddit.com/r/esp32/comments/13t4cf2/i_am_making_board_with_esp32_wroom32d_do_i_have/
    - Source for schematic
        - https://espressif-docs.readthedocs-hosted.com/projects/esp-idf/en/latest/hw-reference/index.html
- RC Circuit for Time Delay
    - According to page 19 of ESP32 data sheet, this is needed to ensure the power supply to the ESP-32 during power-up
- Things to add for UI
    - Emergency Reset switch or button maybe?
    - LEDs that would give user info:
        - PCB is powered on
        - Indicator that TemperatureAir goes below/above threshold
        - Indicator that TemperatureSurface goes below/above threshold
        - Indicator that MoistureSensor goes below/above threshold
- Button
    - Make sure the button is open circuit if not pressed

2/26/2025

- TA was sick, but meet over zoom to discuss questions

    - We can use the dev board during breadboard demo
    - PCB review is optional
    - Order parts through TA to get reimbursed

2/27/2025

- Worked on PCB schematic today, need to do this tomorrow:
    - ADD LED to indicate there is power to PCB
        - Add LED for ON for the sensors?
        - Other LED indicators?
    - ADD test points

3/8/2025

- Add design document stuff
- Switched ESP32 MCU
- PCB mistake, find VIN in the schematic!!! do not put it into 3.3V
- Today working on breadboard demo

3/21/2025

- Plan for modifying PCB of Control Subsystem
    - Find VIN in the schematic, put it into 3.3V
    - Check if the pins match from the previous schematic / PCB design
        - Fix UART pin
    - Check new example schematic
    - Open the pins needed for programming

3/24/2025

- More reminders for modifying PCB
    - Debug pins
    - Find a way to separate power and control subsystem
- Finished updating PCB with ESP32-WROOM-32E
    - https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf
- Added testpoints and cleaned up plane
- Researched how to program ESP32
    - Will try to use UART interface
        - https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/uart.html
    - Will use this to convert USB to UART
        - https://www.silabs.com/interface/usb-bridges/classic/device.cp2102?tab=specs

4/1/2025

- New PCB came in, soldered Control and Sensing Subsystem. Having trouble with placing the decoupling capacitors and hand soldering the ESP32, this took the whole day at least 4 hours.

4/2/2025

- Programmed the ESP32 using the CP2102, connected it to the UART connectors that was set up previously on our PCB.
- Issues with programming, need to add more of the pull up resistors and capacitors. Wanted to be able to program it as least as possible without soldering much, to test if it works
- Started writing software for the initial event loop of the ESP32, need to read the values and serial print it.
- Was able to serial print it
- Initial event loop code:

```
      void loop() {



surfaceTemp = 0;

airTemp = 0;

rainAmount = 0;



surfaceTemp = readSurfaceTemperatureSensor();

rainAmount = readRainSensor();

airTemp = readAirTemperatureSensor();


```

```
/* check if temp sensor is valid value */

if ((surfaceTemp <= -127.0) || (airTemp <= -127.0))

{

  heaterState = false;

  updateSystemState();

  return;

}



/* heater mode is ON */

if (heaterState)

{

  if (surfaceTemp >= surfaceTempOverheatThreshold)

  {

    heaterState = false;
```

```
  }

  updateSystemState();

  return;

}



/* No water detected if true */

if (rainAmount >= moistureThreshold)

{

  heaterState = false;

  updateSystemState();

  return;

}



if ((airTemp < airTempThreshold) || (surfaceTemp < surfaceTempThreshold))

{
```

```
    heaterState = true;

  }

  else

  {

    heaterState = false;

  }

  updateSystemState();

}



float readAirTemperatureSensor()

{

  airTemperatureSensor.requestTemperatures();

  float temperatureC = airTemperatureSensor.getTempCByIndex(0);  // Get
temperature in Celsius
```

```
  Serial.print("Air Temperature: ");

  Serial.print(temperatureC);

  Serial.println(" °C");

  return temperatureC;

}



float readSurfaceTemperatureSensor()

{

  surfaceTemperatureSensor.requestTemperatures();

  float temperatureC = surfaceTemperatureSensor.getTempCByIndex(0);  // Get
temperature in Celsius



  Serial.print("Surface Temperature: ");

  Serial.print(temperatureC);

  Serial.println(" °C");
```

```
  return temperatureC;

}



float readRainSensor()

{

  Serial.print("Raw value of rain sensor: "); // Print information message

  float rain = rainSensor.getRawValue();

  Serial.println(rain); // Prints raw value of rain sensor



  return rain;

}



void updateHeaterState()

{

  if (heaterState)
```

```
  {

    digitalWrite(MOSFET_CONTROL_PIN, HIGH);

  }

  else

  {

    digitalWrite(MOSFET_CONTROL_PIN, LOW);

  }

}
```

4/10/2025

- Decided to make changes with the event loop.
  - Once hazardous conditions are detected, we will ignore everything else and read the surface sensor temperature until it reaches our overheat threshold which is 10C.
  - This is added as a check at the beginning of the event loop after reading the sensor readings

```
/* heater mode is ON */

  if (heaterState)

  {

    if (surfaceTemp >= surfaceTempOverheatThreshold)

    {

      heaterState = false;
```

```
    }

    updateSystemState();

    return;

  }
```

- This will make sure that if the heater State is on, the system will keep heating until the surface temp reaches the overheat threshold before it can start reading again.
- When testing readings from temperature sensors, it was not working. It would output 4095 which means that it is the default value and unreadable.
- After reading the documentation of the DS18B20, it requires a 10k pull up resistor on the data pin https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf
- Need to redo the PCB design to add the pull up resistor, here is the final PCB design *![[Pasted image 20250508141828.png]]
- The pull up resistors have been added and the PCB has been ordered.

4/17/2025

- Final PCB came in, waiting on Kahmil to finish soldering power subsystem
- Soldered Control and Sensing Subsystems and added the 10k pull up resistors that are neccessesary.
- After testing the event loop and adding the temperature values, the ESP32 was able to print out the correct values due to adding the pull up resistors

4/25/2025

- Met with group to test the entirety of our system with a cooler that James bought.
- Talked over initial plans on what to add
    - Waterproofing
    - Using dry ice to get to sub-zero temperatures
    - Functionality on the ESP32
        - Add bluetooth to be able to print out the state of the system and the temperature values as well
        - Add a way to change the thresholds wirelessly
        - Add a way to save data to the csv file?
- Mistakes
    - Unable to use dry ice
        - too expensive
        - cannot get to sub-zero temperatures
- Full code of the code to be flashed onto the ESP32, it basically has functionality to read from the sensors and update the heater State by GPIO 3.3V Output on one of the pins
- There are some bluetooth code in there, but I did not add the bluetooth code.

```
#include <OneWire.h>

#include <DallasTemperature.h>

#include "Simple-rain-sensor-easyC-SOLDERED.h"

#include <driver/adc.h>

#include "Callbacks.h"


#include <BLEDevice.h>

#include <BLEServer.h>

#include <BLEUtils.h>

#include <BLE2902.h>


/* ESP32 Pins */

#define AIR_TEMPERATURE_SENSOR_PIN 19

#define SURFACE_TEMPERATURE_SENSOR_PIN 17

#define RAIN_SENSOR_DIGITAL_PIN 34

#define RAIN_SENSOR_ANALOG_PIN 35

#define RAIN_SENSOR_DRY_VAL 1800

#define RAIN_SENSOR_LED_PIN 21

#define MOSFET_CONTROL_PIN 26



/* LOOP timer */

#define INTERVAL 3000



#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331914b"

#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
```

```
OneWire airTemperatureOneWire(AIR_TEMPERATURE_SENSOR_PIN);
OneWire surfaceTemperatureOneWire(SURFACE_TEMPERATURE_SENSOR_PIN);
DallasTemperature airTemperatureSensor(&airTemperatureOneWire);
DallasTemperature surfaceTemperatureSensor(&surfaceTemperatureOneWire);

SimpleRainSensor rainSensor(RAIN_SENSOR_ANALOG_PIN);
BLECharacteristic* pCharacteristic;
float airTemp, surfaceTemp, rainAmount;
bool heaterState = false;

void setup() {
  Serial.begin(115200);  // Start serial monitor

  /* Initialize the air and surface temperature sensors */
  airTemperatureSensor.begin();
  surfaceTemperatureSensor.begin();

  /* Initialize rain sensor, adc width and pin mode*/
  rainSensor.begin();
  rainSensor.setADCWidth(10);
  pinMode(RAIN_SENSOR_DIGITAL_PIN, INPUT_PULLUP);
  pinMode(AIR_TEMPERATURE_SENSOR_PIN, INPUT_PULLUP);
  pinMode(SURFACE_TEMPERATURE_SENSOR_PIN, INPUT_PULLUP);
  pinMode(MOSFET_CONTROL_PIN, OUTPUT);

  //---BLE Setup---//
  BLEDevice::init("ESP32_BLE");
```

```
  BLEServer* pServer = BLEDevice::createServer();
  BLEService* pService = pServer->createService(SERVICE_UUID);
  // Create a characteristic with read, write, and notify properties.
  pCharacteristic = pService->createCharacteristic(
                      CHARACTERISTIC_UUID,
                      BLECharacteristic::PROPERTY_READ |
                      BLECharacteristic::PROPERTY_WRITE |
                      BLECharacteristic::PROPERTY_NOTIFY
                    );
  pCharacteristic->addDescriptor(new BLE2902());
  pCharacteristic->setCallbacks(new Callbacks());
  pService->start();
  BLEDevice::getAdvertising()->start();
  Serial.println("BLE Server started and advertising...");
}

void loop() {

  surfaceTemp = 0;
  airTemp = 0;
  rainAmount = 0;

  surfaceTemp = readSurfaceTemperatureSensor();
  rainAmount = readRainSensor();
  airTemp = readAirTemperatureSensor();

  /* check if temp sensor is valid value */
```

```
if ((surfaceTemp <= -127.0) || (airTemp <= -127.0))
{
  heaterState = false;
  updateSystemState();
  return;
}

/* heater mode is ON */
if (heaterState)
{
  if (surfaceTemp >= surfaceTempOverheatThreshold)
  {
    heaterState = false;
  }
  updateSystemState();
  return;
}

/* No water detected if true */
if (rainAmount >= moistureThreshold)
{
  heaterState = false;
  updateSystemState();
  return;
}
```

```
  if ((airTemp < airTempThreshold) || (surfaceTemp < surfaceTempThreshold))

  {

    heaterState = true;

  }

  else

  {

    heaterState = false;

  }

  updateSystemState();

}



float readAirTemperatureSensor()

{

  airTemperatureSensor.requestTemperatures();

  float temperatureC = airTemperatureSensor.getTempCByIndex(0);  // Get
temperature in Celsius



  Serial.print("Air Temperature: ");

  Serial.print(temperatureC);

  Serial.println(" °C");

  return temperatureC;

}



float readSurfaceTemperatureSensor()

{

  surfaceTemperatureSensor.requestTemperatures();

  float temperatureC = surfaceTemperatureSensor.getTempCByIndex(0);  // Get
temperature in Celsius
```

```
  Serial.print("Surface Temperature: ");
  Serial.print(temperatureC);
  Serial.println(" °C");
  return temperatureC;
}

float readRainSensor()
{
  Serial.print("Raw value of rain sensor: "); // Print information message
  float rain = rainSensor.getRawValue();
  Serial.println(rain); // Prints raw value of rain sensor

  return rain;
}

void updateHeaterState()
{
  if (heaterState)
  {
    digitalWrite(MOSFET_CONTROL_PIN, HIGH);
  }
  else
  {
    digitalWrite(MOSFET_CONTROL_PIN, LOW);
  }
}
```

```
void sendValues() {


  char buffer[32];
  sprintf(buffer, "%.2f;%.2f;%.2f; %.2f", surfaceTemp, airTemp, rainAmount,
(float)heaterState);
  String valueString = String(buffer);



  // Set and notify the new value.
  pCharacteristic->setValue(valueString.c_str());
  pCharacteristic->notify();
  Serial.println("Sent values...");
}


void updateSystemState()
{
  updateHeaterState();
  sendValues();
  Serial.println("");
  delay(INTERVAL);  // Wait 2 seconds before reading again
}
```

4/26/2025

- Added functionality to save CSV files using python csv library
- Did some more testing and getting test data for presentation
- Notes for each component on Control Subsystem and Sensing Subsystem and what they do
    - ESP32-WROOM
        - Chosen because of its wireless and bluetooth capabilities

        - This is useful because our system will be in a cooler to simulate rigid temperatures, we dont want to keep opening it and closing
    - DS18B20 Temperature Sensor
        - Chosen because it is rated for +- 0.5 C accuracy
        - Waterproof
        - 3.3V Power
        - We will have two sensors, one air temperature and one surface
    - Rain Sensor
        - Chosen because it can detect how much water is on the sensor
        - Returns 0-4095 value
        - Due to testing, we decided that 1800 is our threshold for the water level

4/27/2025

- More testing and getting test data
    - Testing takes a while since we have to wait 5 minutes to get the temperatures low enough
    - Another 5 minutes to heat up the bridge
    - Then we have to wait for temperatures to go back to normal when the heaters turn off
    - This is just for one iteration
- Images of fully ran tests
    - ![[Pasted image 20250508144816.png]]
    - ![[Pasted image 20250508144837.png]]
    - ![[Pasted image 20250508144846.png]]
    - ![[Pasted image 20250508144859.png]]
    - These images will be used during the presentation for the data analysis

4/28/2025

- Did the final demo

5/3/2025 -5/4/2025

- Practiced presentation, did the slides
- Borrowed sam's heatgun to show uniformity in terms of heat
    - ![[Pasted image 20250508144953.png]]
- Dry runs