# Using Temperature and Humidity Sensors to Propose a New Form of Flat Roof Leakage Detection

Francis Robson

29/04/2020

# Contents

## List of Figures

## 1 Abstract

The research highlights several existing methods for flat roofing leakage detection. The majority of implemented leakage detection methods are inefficient active methods which require regular testing; passive methods, while more efficient, are costlier to implement. In the first section of the paper, these individual methods are highlighted and compared with one another. From this, several hypotheses about the behaviour of moisture within flat roofing insulation are formulated, and these are used to formulate several research questions. These research questions are used to design experiments to investigate these behaviours of roofing insulation further, and these experiments and their findings are presented in the next section. From these experimental results, a model is proposed which aims to use the findings to passively locate roofing leakages more effectively and in a more cost-efficient manner, based predominantly around the behaviours in change of humidity within an insulation block. Conclusions are then drawn, and further works required to fully implement this model, as well as any possible improvements suggested, are then discussed. It is hoped the research can contribute towards a greater understanding of the behaviours of roofing insulation, and assist with the development of a more cost-efficient and accurate passive leakage detection system.

## 2 Introduction

For the majority of human history, pitched roofing was quintessential with the majority of human architecture. This was largely out of necessity rather than stylistic choice, as for many years, drainage technology was simply not equipped to deal with adverse weather conditions, and thus sloped roofing was used in order to allow gravity to drain rainwater naturally. However, in recent centuries, drainage technology has advanced to the point wherein flat, or more commonly slightly pitched, roofing is possible. Indeed, as globalisation spreads, flat roofing has become a necessity in commercial buildings, for several key reasons. The first of these is the crowding of urban areas which accompanies the sharp upturn in population growth in the past century; this has led building developers to increase the average height of buildings, as the horizontal land mass is quite often simply not available. Secondly, is the complexity of ventilation of buildings of this size; HVAC (Heating, Ventilation and Air Conditioning) systems are required in order to maintain safe air standards for commercial and public buildings, and the central units of all these systems require housing in a well-ventilated area. As such, the common practise has simply become to house these central units on the roof areas of any large commercial building; this is why flat roofing has become synonymous with commercial, industrial, and other public buildings, while the standard for residential housing remains largely pitched or sloped.

However, as previously mentioned, drainage of a flat surface presents its own unique difficulties; water may remain present on a flat surface for significantly longer than it would remain on a pitched or sloped surface, which presents a much greater strain on the roof coverings, insulation, and any other components of the roofing. Commonly, the load-bearing of flat roofing will be undertaken by structural joists above the ceiling of the highest floor. Above this will be a decking of some kind of material (largely either wooden or metal in construction), a water retardant layer, and a layer of insulation, above which the various layers of waterproof membrane or other roofing material will be attached. An example of this layout can be seen in Figure 1.

While the waterproof membrane is intended to be fully waterproof, a variety of different factors can negatively influence this. For example, human error in application of the membrane, wear from a variety of different weather conditions, or loosening adhesive, could all lead to breaches within the membrane's surface. However, these breaches may often be invisible to the naked eye, and are often housed on the roof of large buildings, which are very seldom inspected for such leaks. This may lead to breaches going undetected for long periods of time, allowing moisture from weather conditions to seep through the insulation material, which will be porous in nature. If moisture is allowed to reach the decking, joists, or ceiling, this can manifest as damp, leading in extreme cases to potential structural damage, or leaks into the floor below. Repairs of this type of damage are often very costly, as in order for these repairs to be carried out, all upper layers must be removed, and then subsequently replaced once the repairs are

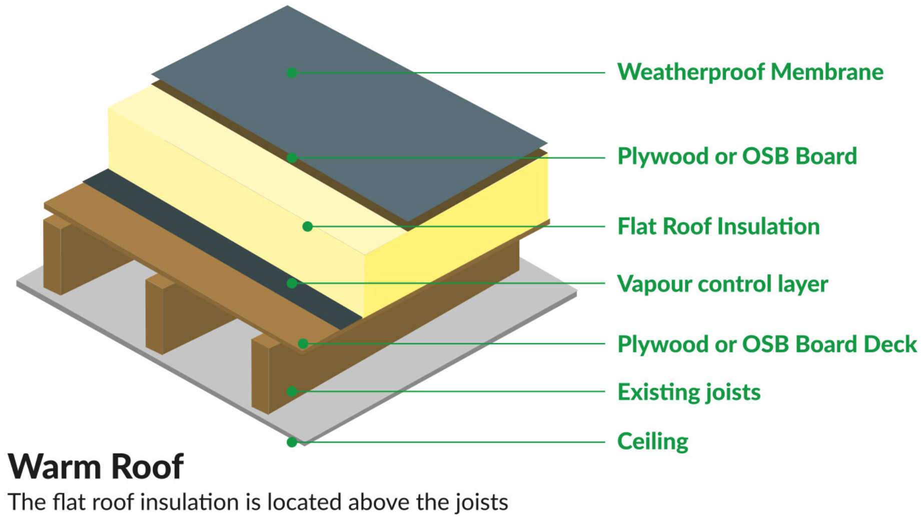


Figure 1: Layers of a Flat Roof [2]

completed. As such, there is significant demand for any kind of leak detection systems, which may alleviate significant repair costs in the future. For example, [1] over a 9 year period on a large university building complex, found their average roofing replacement costs to be approximately $30/square metre over that time. However, this cost was largely concentrated in one period of time, with $2.6million out of a total $4.5million spent across a 2 year span, when full replacement works rather than simple repair works were required. The study also found that each building could expect approximately 5.2 leakage incidents per year, and the insufficient warning was what required these much more costly replacement works.

This paper aims to analyse the currently existing forms of active and passive flat roof leakage detection systems, and then propose and test a form of flat roof leakage detection based upon humidity and temperature sensors, through analysing and modelling moisture's effect on flat roof insulation. System requirements will be analysed, a hardware and software system will be designed and built, data collected, results analysed, a model constructed, and conclusions drawn and future works proposed.

# 3 Background & Current Works

Currently existing forms of leakage detection systems for flat roofing and flat roofing insulation take one of two forms; active or passive detection systems. Active systems rely on a building owner proactively and regularly searching for leaks, with little to no prior knowledge of whether a search may be successful. Passive systems instead alert the building owner whenever a leakage is detected, and thus are much more desirable. However, the majority of existing systems take the form of active systems, as all passive systems consist of significantly more recent technology, wherein costing is still very much an issue. This section will assess existing passive and active leak detection systems, and use them to propose the research questions that will inform the remainder of this paper. A comparison of each detection method can be seen in Table 1.

## 3.1 Active Measurement Methods

### 3.1.1 Flood Testing

Flood testing is the simplest form of testing for a breach in the membrane: all drains available to the roof surface are closed and the roof is temporarily dammed off, then water is added to the surface and retained there for 24 hours, after which the below surface is inspected for any signs of leakages, as seen in Figure 2. The principle behind this method is that such a prolonged period of time exposed to continuous water pressure will expose any breaches, as this will immediately become the path of least resistance for the water to follow. This method has several obvious disadvantages, principally the difficulty in transporting large volumes of water to a rooftop, especially of a high building, then removing it again, as well as the inconvenience of having to block all drainage outlets and render the roof inaccessible for a 24-hour period. Additionally, there is no more reliable method of discovering leaks at the close of the 24-hour period than the human eye, so it can still be deeply unreliable. [3, 4, 5]

Figure 2: Flood Testing in Progress [3]

### 3.1.2 Low Voltage Testing

Low voltage testing is one testing method based on the principle that many roofing membranes are water-retardant as well as electrical insulators, so in theory a fully intact roofing membrane will allow no electrical current to flow through it. However, in the case that there is a breach, the water present through the insulation will act as an electrical conductor, completing the circuit between the two sides of the membrane. In order to quantifiably measure this, a small depth of water is applied across a roof membrane, which then has a small current applied across it. The current is grounded to the roof decking below the insulation, so in order for this method to be undertaken, the roof decking must be an electric conductor. This principle is shown in Figure 3. Through taking a series of measurements, the operator can pinpoint locations on the roof membrane where current has been allowed to flow, and from here deduce or find the exact locations of the breaches and note them for repair. However, there are still flaws with this method of testing; it does not give exact results as to where a breach is located, only results for an operator to analyse, thus putting the method to be only as successful as the operator is proficient. Additionally, in scenarios where a breach is recent, there is a possibility that water has not fully permeated through to the decking; in these scenarios, this test will not return any results, so it does not have a fully successful capture rate. [3, 6, 7]

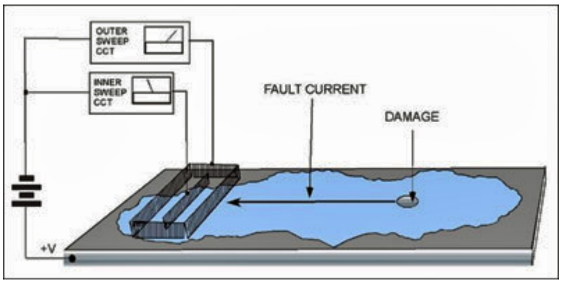


Figure 3: Low Voltage Testing Schematic [3]

### 3.1.3 High Voltage Testing

High voltage testing operates on exactly the same principles as low voltage testing, wherein the decking must be an electrical conductor, and the roof membrane must be an electrical insulator, and only in the presence of the breach can a current flow between the two, as seen in Figure 4. However, instead of a low voltage being applied across the entire membrane, instead a high voltage handheld appliance is run across the membrane in vertical lines, while also again being grounded to the conducive decking below, an example of which is shown in Figure 5. In the scenario that a current is allowed to flow, the device is programmed to immediately cut off the voltage and alert the test operator of the breach. Having made a note of the vertical co-ordinate of the test succeeding, the operator will then scour the membrane in horizontal lines, with any intersections being noted down as breach co-ordinates. This method is distinctly more convenient than low voltage testing due to the single handheld appliance required with no external water having to be applied; however, it suffers from several of the flaws low voltage testing demonstrates, namely the reliance on the individual operator's skill, as well as a breach having permeated fully in order to be detected. [3, 8]

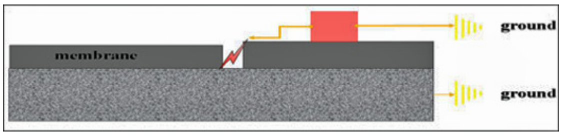


Figure 4: High Voltage Testing Schematic [3]

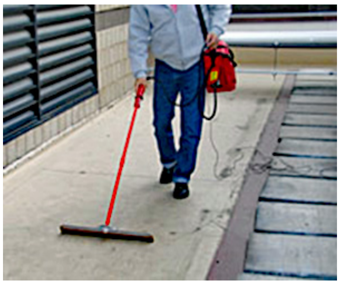

Figure 5: High Voltage Testing Equipment [3]

### 3.1.4 Thermal Imaging

Thermal imaging is one of the few active detection methods which does not rely on the principle of electric conductance. Instead, it relies on the property of heat transfer; water is slower to transfer thermal energy than air, so upon surverying an area of roof insulation through an infrared camera, any patches of moisture located within the insulation will show up at a cooler temperature than the rest of the roof. Alternately, by surveying the roof at the end of the day after the sun has been shining, any patches of moisture will show up as warmer than the surrounding areas, due again to the slowed rate of energy transfer. Examples of the images produced by this can be observed in Figures 6 & 7. However, this method suffers from much the same issue as the other listed active measurement methods, in that it only provides the operator with measurements that they must decipher individually. Additionally, this method only locates pockets of moisture; it does not account for any kind of possibility that the water is within the insulation layer for a reason besides a breach in the roof membrane. [3, 9]

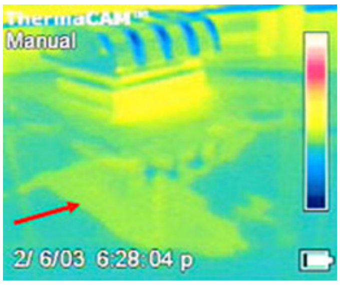


Figure 6: Roof Viewed Through Thermal Imaging [3]

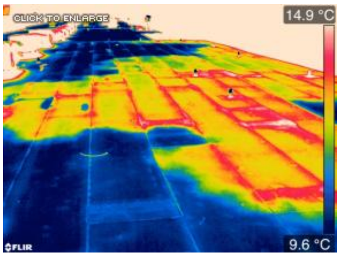


Figure 7: Roof Viewed Through Thermal Imaging 2 [10]

#### 3.1.5 Capacitance Testing

Capacitance testing again works upon the principle that moisture will be conducive through a membrane, however it does so through somewhat different methods to high or low voltage testing. In capacitance testing, a small, handheld device emits a low-frequency electronic signal, measuring the area below's conductivity; comparisons with known 'dry' and 'wet' areas of the same materials can lead to accurate assumptions about whether moisture is present in the roof. Again, there are several flaws to this active measurement system; it is impossible to carry out while a roof membrane is wet due to the inherent conductivity present in surface water, and accurate readings are only present when roof materials and layers are of a uniform distribution and thickness. Additionally, measurements are carried out with a small handheld device, with a greater number of readings leading to a more coherent measurement picture, leading to a time-consuming measurement process, which is again heavily reliant on the skill of the operator. [3, 11]

## 3.2 Passive Measurement Methods

### 3.2.1 Embedded Moisture Sensors

Embedded, passive measurement methods to detect leaks without the need for active measurements are very recent developments, with the majority of technology being developed in the past 10 years. [12, 13] is one such development, utilising the principles of low voltage testing to create a grid of conductive tape embedded within the roof, on the vapour control layer, to continuously monitor the roof for any signs of electrical conductivity. The grid network can pinpoint the location of the breach to within a reasonable degree of accuracy, however, the major downside to this method is that it requires a continuous power source and a great deal of wiring; while this may be a suitable method for very large scale operations, it may be too costly or inefficient for smaller buildings. [14] presents an inductive-capacitive resonant circuit, implanted within a desired hard-to-reach area such as behind a drywall. The presence of water vapour in the air surrounding the sensor increases the capacitance of the spiral inductor, reducing the sensor's resonant frequency, which is being monitored from the other side of the wall. The major downside of this method is it only monitors a very specific location; while it may be suitable for within a wall, for across a wide space of roof it would be a great deal less effective.

[12, 13, 14] indicate that there is some success to be potentially found through investigating the thermal properties of insulation. The founding principle of Thermal Imaging is the tenet that areas of insulation containing greater concentrations of moisture will tend towards a cooler temperature. Additionally, voltage and capacitance testing rely on the principle that additional moisture present in an insulation block will increase the potential for electrical current to flow. As humidity is simply defined as a measurement of moisture content, the hypothesis presided over in this paper is that a knowledge of temperature and humidity readings within a block of insulation can be used to formulate accurate predictions about the presence of a leakage in the roofing membrane above that insulation block.

### 3.2.2 Comparison of Measurement Methods

| **Measurement Name** | **Active or Passive** | **Pros** | **Cons** |
|---|---|---|---|
| Flood Testing | Active | - Quite straightforward concept | - Lengthy process<br>- Reliant on operator<br>- Unreliable results<br>- Difficult to transport water to and from test<br>- Long setup required |
| Low Voltage Testing | Active | - Requires relatively little setup | - Does not detect if not fully permeated<br>- Lengthy process<br>- Roof membrane must be insulator<br>- Roof decking must be conductor |
| High Voltage Testing | Active | - Relatively fast process<br>- Only requires handheld device<br>- Requires no setup | - Does not detect if not fully permeated<br>- Roof membrane must be insulator<br>- Roof decking must be conductor |
| Capacitance Testing | Active | - Only requires handheld device<br>- Requires no setup | - Cannot be used on wet surfaces<br>- Reliant on operator<br>- Does not detect if not fully permeated |
| Thermal Imaging | Active | - Non-intrusive<br>- More accurate detection of water<br>- Relatively fast process<br>- Requires no setup | - Results still require processing<br>- Does not account for why water would be present |
| Embedded Moisture Sensors | Passive | - Does not require operator<br>- Continuous measurements over large area<br>- More reliable results | - Require roof to be electrified<br>- High cost<br>- If damaged/faulty, roof must be replaced to fix |

Table 1: Comparison of Leak Detection Methods

# 4 Research Hypotheses & Questions

## 4.1 Technical Problem Statement

This thesis aims to propose a low-energy, passive leakage detection system for flat, commercial roofing, making use of readily available sensor technology.

Current detection technologies are either active and too reliant on operator ability and regular checks, or are passive and have much greater cost and energy costs. Through this, many leakages are missed, and preventable amounts are spent on roofing repairs and replacements each year.

Through evaluating appropriate sensor readings and measurements, this thesis will establish whether temperature and humidity readings are suitable as a passive sensor network to accurately monitor roof membrane leakage.

## 4.2 Hypotheses

Currently existing methodologies reviewed in Section 3 imply several hypotheses regarding the behaviour of moisture within flat roofing insulation, which should be ratified into any new leakage detection system. These core hypotheses are listed below, and used to formulate the key research questions listed in Section 4.3:

- 1) Humidity and Temperature remain constant through a block of flat roofing insulation in relation to ambient outdoor temperatures when no moisture is present within the block, as is the founding principle behind all the reviewed testing methods.
- 2) Humidity within a block of flat roofing insulation increases to a set maximum the closer measurements are to a leakage entry point, as implied by Thermal Imaging (3.1.4).
- 3) Temperature is inversely proportional to humidity, as implied by Thermal Imaging (3.1.4).
- 4) Moisture disperses evenly in all directions within a block of flat roof insulation when block is completely flat, as implied by Flood Testing (3.1.1).
- 5) Moisture disperses at a constant rate from a leakage entry point within a block of flat roof insulation; this is slower than the rate of dispersal through air,as implied by Flood Testing (3.1.1).

## 4.3 Research Questions

- 1) How suitable are temperature and humidity sensors for leakage detection?
  - What is the temperature and humidity distribution model throughout a block of insulation?
  - How does the presence of moisture within an insulation block impact the temperature and humidity distribution model?
- 2) If temperature and humidity sensors are suitable, where should these sensors be located?
  - How does distance from moisture affect sensor readings?
  - How does depth within a block affect sensor readings?
- How suitable is this model for a future leakage detection system?
  - Within the model, how should sensors be placed in order to eliminate areas of no coverage?
  - What future works and improvements are necessary for such a system?

## 5 Experiment Design

In order to validate the proposed hypotheses and answer the research questions formulated in Section 4.3, a system of temperature and humidity sensors must be designed and assembled. Data which highlights the thermal characteristics within the blocks of insulation must be collected, with respects to control variables such as distance from entry point, depth, and time. This section details the design process of these experiments, with regards to choice of hardware, software design, and experiment strategy, as well as detailing any issues encountered through the design and construction phases. Section 5.1 highlights some formulated System Requirements.

For these experiments, 30x30x8cm blocks of 'Recticel Insulation Eurothane GP PIR Insulation Board', hereafter referred to as PIR, will be used. Traditionally, insulation comes in 60x60cm blocks as an industry standard, however in order to reduce overall experiment duration, the dimensions of each block were halved, so as to increase the amount of material available. This decision is based upon the assumption that as roofing insulation is a highly standardised industry, a smaller block would still retain the same attributes as a larger block, as internal characteristics should be close to identical, and measured attributes will scale in a linear fashion.

PIR, or polyisocyanurate, is a thermosat plastic, which is commonly produced as a foam block and used as rigid thermal insulation in the construction industry [15].

Two different sets of 5 groupings of 3 sensors, '12345' and '16278' were decided upon to go within the insulation block. The locations of these sensors with relation to the insulation block can be seen in Figure 8. These locations were decided on in order to allow the most variety with distance from moisture entry point within data collected. Within each grouping of 3 sensors, a sensor will be located at 3cm, 5cm and 7cm from the surface of the insulation block. 6 different moisture entry points will be used, also shown in Figure 8; points e1 - e6. Using both sets of sensor placements for each moisture entry point will result in 12 different data sets, with time, horizontal distance and vertical distance being used for relative measurement.

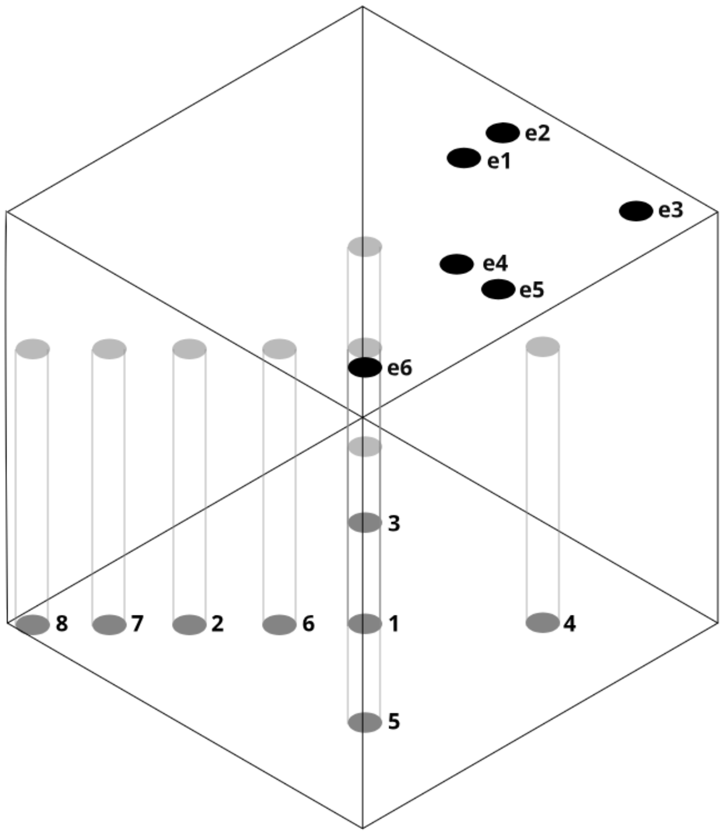


Figure 8: Sensor & Entry Point Layout

## 5.1 System Requirements

There are several categories of System Requirements. Firstly, hardware requirements detail the physical capabilities of the designed system, while the mechanical requirements detail the physical properties of a chosen setup. The software requirements detail the required capability of any software developed for the testing setup.

### 5.1.1 Hardware Requirements

- Capable of reading 15 digital temperature/humidity sensors concurrently.
- Capable of data transmission for all sensor readings.
- Operating Humidity measurement range of 0-100%.
- Operating Temperature measurement range of 0-50°C.
- Operating accuracy error of ±5% maximum.

- Capable of sampling at a suitable sampling frequency (estimated at 0.2Hz or slower)

#### 5.1.2 Mechanical Setup

- 15 sensors can fit within a 30x30cm block of roofing insulation.
- 15 sensors can be connected to a single microcontroller.
- All hardware components fit within a budget of 200 EUR.
- All hardware components within the block can withstand expected humidity and moisture levels (estimated up to 80%humidity).

#### 5.1.3 Software

- Capable of reading all transmitted data streams.
- Capable of data writing to permanent storage for purposes of analysis.
- Capable of splitting stored data into meaningful components.

### 5.2 Hardware Setup Design

#### 5.2.1 Humidity & Temperature Sensors

The sensors chosen for these experiments were the Adafruit AM2302 Humidity & Temperature Sensors, as seen in Figure 9.

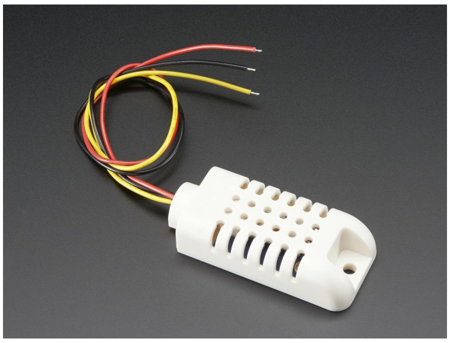

Figure 9: Adafruit AM2302 Sensor [16]

These were chosen due to their suitability with regards to the Hardware and Mechanical System Requirements, as outlined in Section 5.1:

- Cost: 15 sensors can be purchased within the specified budget, allowing for additional purchase of a microcontroller to power the sensor network.

- Size: 15 sensors can fit comfortably within a block of 30c30cm block of roofing insulation.
- Wiring: These sensors have 1 digital output, so can be connected to a single microcontroller.
- Humidity Measurement Range: Humidity measurement range of 0-100%.
- Temperature Measurement Range: Temperature measurement range of 0-50°C.
- Sampling Frequency: Capable of sampling at rates of 0.5Hz or slower.
- Durability & Housing: The AM2302 is a DHT22 humidity & temperature sensor, contained within a semi-waterproof protective chassis with extruding wires, with the required pullup resistor also housed.

### 5.2.2 Experiment Housing

A housing was constructed of several layers of laser cut fibreboard and sealed with wood glue, while mountings for the sensors to maintain their depth within the insulation were 3D printed from Acrylonitrile Butadiene Styrene (ABS). Design models for the housing can be seen in Figure 10, and for the sensor brackets in Figure 11. In order to fit the sensors and their mountings, holes were cut in the lower half of the blocks of insulation, so it can be placed directly over the top of the installation, with the moisture inserted from above.

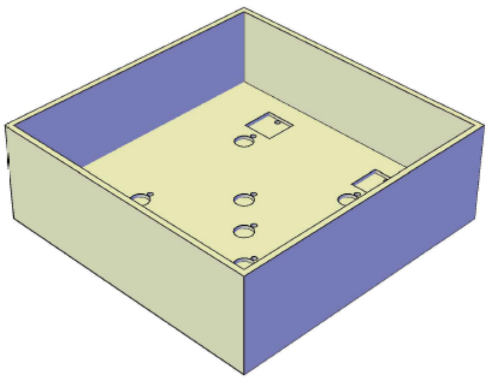

Figure 10: 3D Housing Design

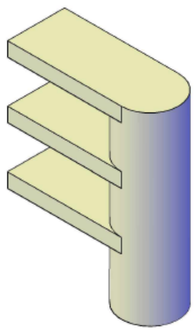

Figure 11: 3D Mounting Design

### 5.2.3 Microcontroller

In order to facilitate the concurrent reading of 15 temperature and humidity sensors, an Arduino Mega 2560 Microcontroller was chosen. This complies with the formulated System Requirements, as outlined in Section 5.1:

- Cost: Alongside the 15 humidity & temperature sensors, fits within the specified budget of 200 EUR.
- Capable of reading 15 data streams concurrently, as the microcontroller has over 15 digital input/output ports.

### 5.2.4 Full Hardware Setup

A schematic of the setup of the connections for each individual sensor (to be repeated for all 15 AM2302 Sensors) can be seen in Figure 12. A picture of the full setup, including the experiment housing and all wiring, can also be seen in Figure 13.

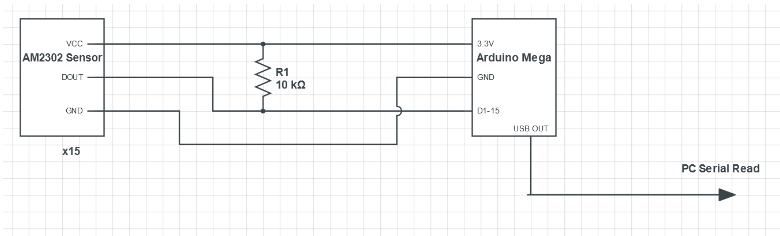


Figure 12: Hardware Setup Schematic

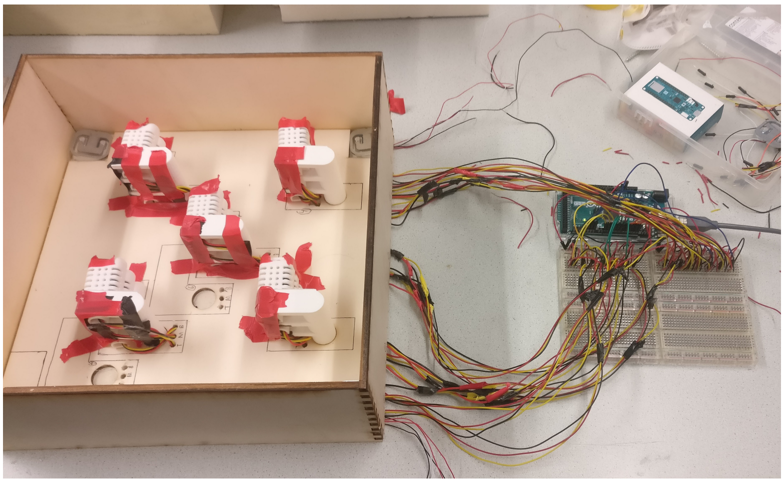

Figure 13: Full Hardware Setup, Including Wiring

## 5.3 Software Setup

With a hardware setup, it was also necessary to assess requirements for the software aspect of the experiments. The software is divided into two areas. The first area is the microcontroller software, which is responsible for accurate timing, reading the sensors, and relaying all data through the USB Serial port. The second area is responsible for monitoring the USB Serial Port, determining whether any data is being received, and if so writing the received data into a .txt file to be analysed later. A second aspect to the second area was later determined to be required, which would break the whole .txt file down into a multitude of single data stream files, to allow for faster analysis.

### 5.3.1 Microcontroller Software

The Microcontroller software was written in the Arduino IDE, largely using the public domain "DHT.h" library. This provides functions to read the temperature in both Fahrenheit and Celsius, read the humidity as a percentage, and to compute the 'Heat Index', which is the temperature perceived by the human body, taking humidity's effect into account. The software reads all 15 sensors at intervals of 5 seconds, meeting the system requirements outlined in Section 5.1.1, taking the humidity, temperature and heat coefficient readings. It uses two arrays to calculate the differential between the current reading and the previous reading, and then prints the readings of Temperature, Change of Temperature, Humidity, Change of Humidity, and Heat Coefficient over the USB Serial Port. The full code can be seen in Appendix A. A basic representation of the software's functionality can be seen in Figure 14.

### 5.3.2 Serial Reading & Writing

The Serial receiving software was written in Python, due to its meeting of the software system requirements outlined in Section 5.1.3. Through the 'serial' library, communications with the USB Serial port could be established, and Python allows for simple programs with the ability to read and write text files. The program would monitor the Serial port for incoming data, decode the incoming string from binary to a readable UTF-8 format, and then write this decoded string to a specified .txt file. If the notifier '7205000ms' appears, this is equivalent to 2 hours in milliseconds, so the program then terminates. The code can be seen in Appendix B.

After the experiments had been written into .txt files, a second program was required to separate the 5 written strings by sensor and by type of reading. A program which would open a specified file and break it into the individual composite pieces, sorted by sensor location and reading type, can be seen in Appendix C. This returns individual sensor readings, which can be fed into another program for analysis and modelling.

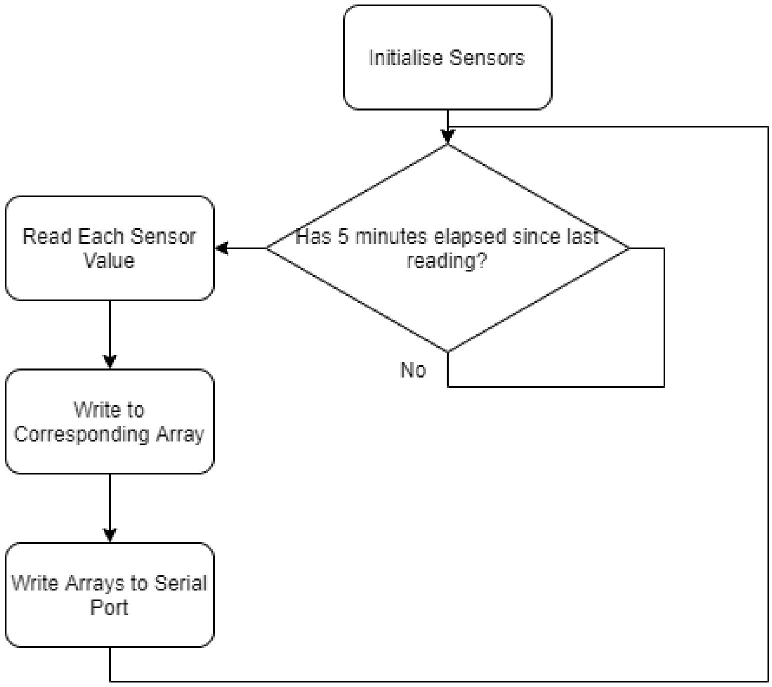


Figure 14: Microcontroller Software Flowchart

### 5.4 Outline of Experiments

12 experiments were conducted in the data acquisition phase. 6 experiments were each conducted with the sensor layouts '12345' and '16278', as laid out in Section 5. Each of these consisted of a 2 hour experiment duration, wherein 10ml of water was inserted at one point of 'e1' to 'e6', as laid out in Figures 15 & 16, in a 30mm hole, with any excess water remaining in a funnel above the surface to trickle down throughout the experiment. It was deemed necessary to use such a minimal amount of water within a drilled hole due to PIR's high levels of water retardation, in order to artificially accelerate the permeation process to reach a level where noticeable metrics could be observed. After each experiment, the block was drained of any residual water and dried, and a dry block was used for the next experiment. In order to maintain controlled conditions, all experiments were conducted in a room at a controlled temperature of 21.5°C, and in between all experiments the housing and sensors were manually dried and allowed to return to ambient temperature.

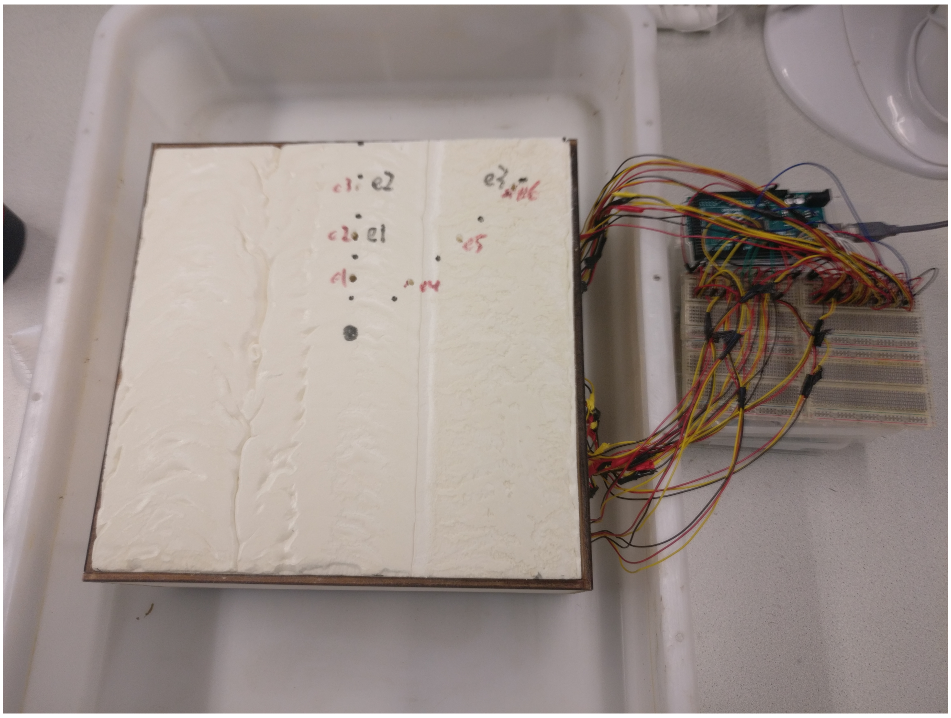


Figure 15: Placings e1-e3 (Black Text)

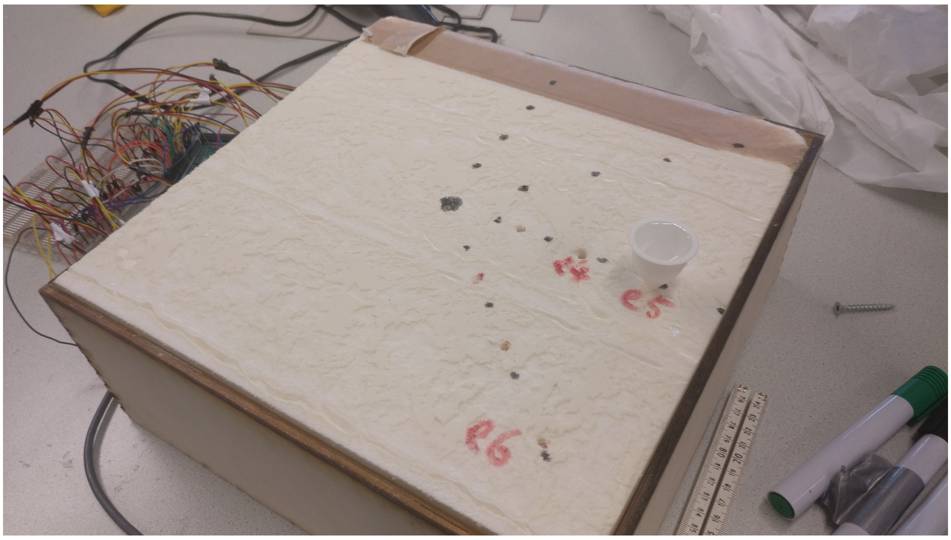


Figure 16: Placings e4-e6, with Funnel (Red Text)

### 5.5 Difficulties Encountered

After the experimental phase had been completed, it was noted that in some final data sets, a lack of computational power had resulted in a series of missed readings and writings, either through taking too long to complete a writing stage, or for another unknown reason. Unfortunately, due to significant timing constraints, it was impossible to redo the experiments, as the errors were discovered later on in the data analysis section. However, the issue was at the computer end of the Serial communications, so all data that was written was otherwise correct, however significant interpolation must be used when observing trends with these data sets. In future experiments, the 'print' line of the Python Serial code would be removed to solve this issue.

# 6 Experiment Results

## 6.1 Data Analysis & Trends

Once all experiments had been completed and data separated according to sensor grouping, moisture entry point and individual sensor, these data collections were fed individually into Microsoft Excel in order to easily generate graphical representations of the data, to visualise trends and identify any correlations, and identify the relationships between readings. For purposes of comparison, figs. 17 to 25 show the ambient readings of the '12345' sensor group at the 3 different measurement heights, while figs. 26 to 34 show the same sensor groupings and positionings, with moisture entry at point 'e2'.

It is now possible to refer to the 5 hypotheses formulated in Section 4.2, and confirm whether these hypotheses can be considered validated. Firstly, viewing the ambient readings, humidity and temperature appear to remain close to constant without the presence of moisture, confirming hypothesis 1. Secondly, a large increase in humidity appears to correlate directly to a decrease in measured temperature. This aligns with the principles of heat transfer outlined in Section 3.1.4, and with formulated hypothesis 3. As the water content in the air present around the sensors increases, the transfer of heat through those air particles slows, leading to a decrease in measured temperature. Thirdly, the sensors closer to the entry point of the moisture, appear to demonstrate a faster increase in humidity, as hypothesized in hypothesis 2. Finally, as expected from a material of uniform characteristics, the outwards dissipation of moisture which is measured through an increase in humidity at a sensor location, appears uniform regardless of direction from entry point; as hypothesized in hypothesis 4. With the presence of a slight incline, it would be expected to skew results slightly.

Additionally, some anomalous results occurred, however these can also be explained through the predicated hypotheses. The humidity values for sensor grouping '1' visible in figs. 26, 29 and 32 increase at a significantly faster rate than the other sensor groupings, however the rate of increase quickly slows to be in line with the other sensor groupings. It is likely that some of the moisture accidentally entered through another hole in the insulation, directly above the '1' sensor grouping, at the beginning of the tests, causing a spike in humidity readings which later stabilised with expected growth in readings. This suggests the validation of hypothesis 5, as the rate of dispersal through the block remained uniform, and only changed noticeably when accidentally allowed to travel through air. In figs. 29 and 32, it can also be observed that towards the end of the 2 hour period, particularly on the low sensors, the rate of change transforms from a logarithmic rate of change to a linear increase. It is possible that some moisture had travelled through a hole in the insulation indirectly, and had saturated the base of sensor grouping 4. One suggested area of future works is the effect saturation has on rate of change, and if this is the expected effect. If so, this further emphasises the need for early detection systems; such a rapid increase in humidity would hugely facilitate damp, increasing the prospect of property damage were this state to be reached.

Observing the results obtained for temperature readings across the different sensors (figs. 18, 21 and 24), much greater levels of fluctuation, possibly caused by external changes in temperature, can be seen. These fluctuations present enough discrepancy in the data, even with relatively small changes in external or ambient temperatures, that the answer to the proposed research question in Section 4.3 'how suitable are temperature sensors for leakage detection' must be considered that they are insufficient. This is theorised to be due to the significantly more complex relationship between temperature and moisture, as the relationship between change in temperature can only be predicted accurately with much greater control over the internal and external temperatures, which is infeasible for any roofing system. Greater detail in readings, with additional data measuring external temperature as well as internal, may be required in order to accurately model temperature behaviour within roofing insulation. However, the results for humidity readings suggest that a model for humidity change can be derived from the findings. This model is expanded on in Section 6.2.3.

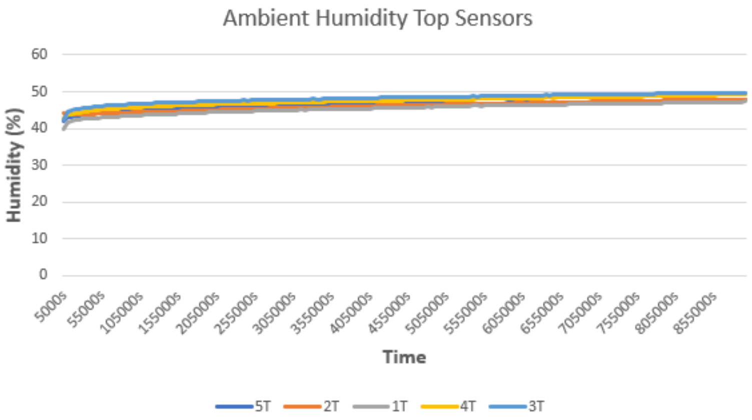


Figure 17: Ambient Top Humidity Readings

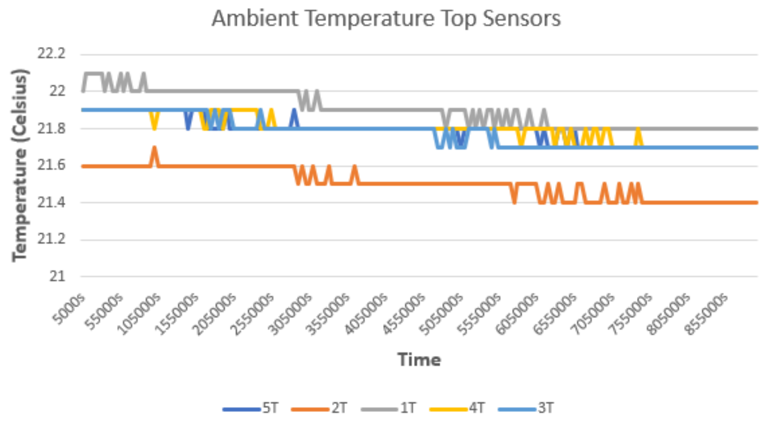


Figure 18: Ambient Top Temperature Readings

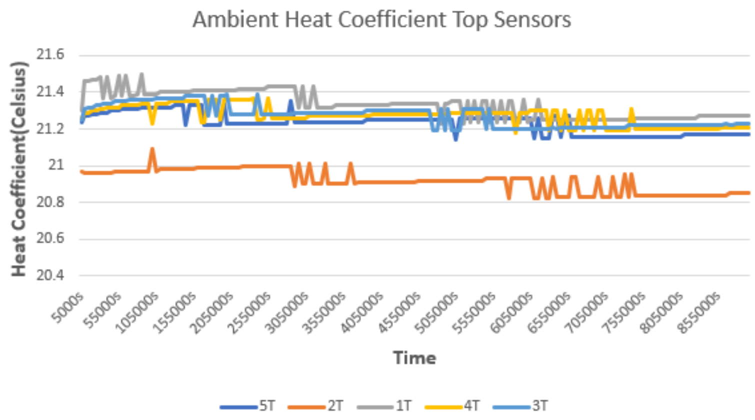


Figure 19: Ambient Top Heat Coefficient Results

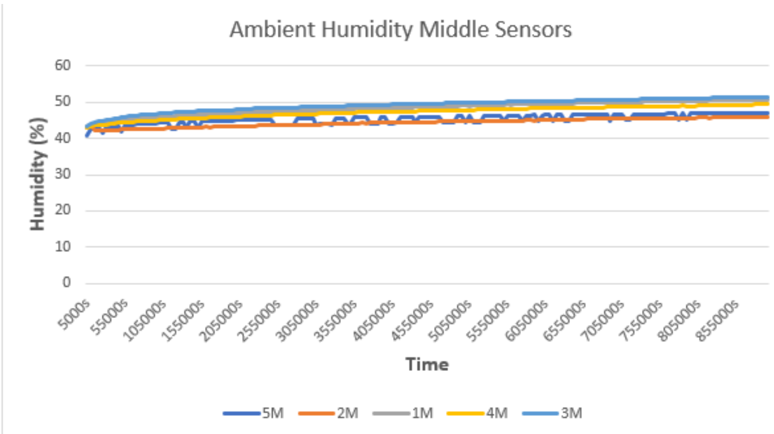


Figure 20: Ambient Middle Humidity Readings

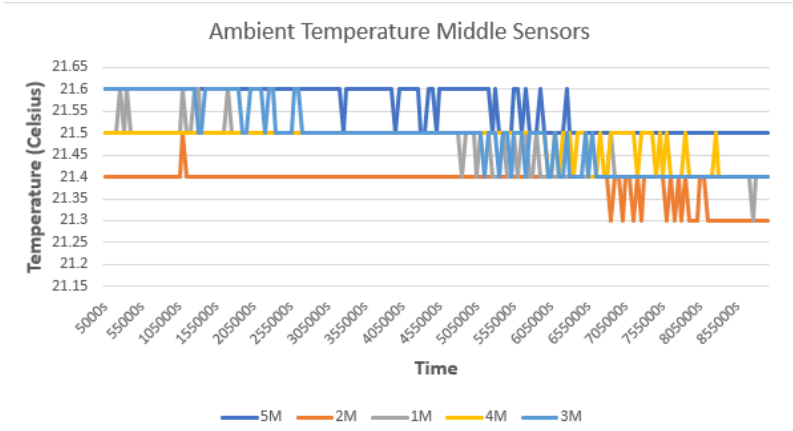


Figure 21: Ambient Midldle Temperature Readings

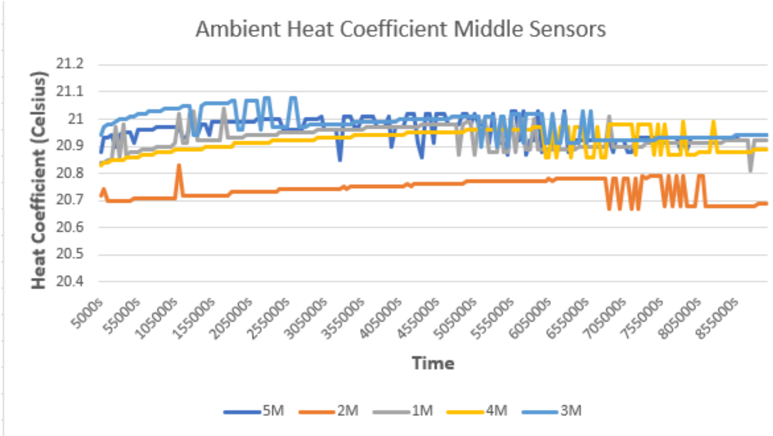


Figure 22: Ambient Middle Heat Coefficient Results

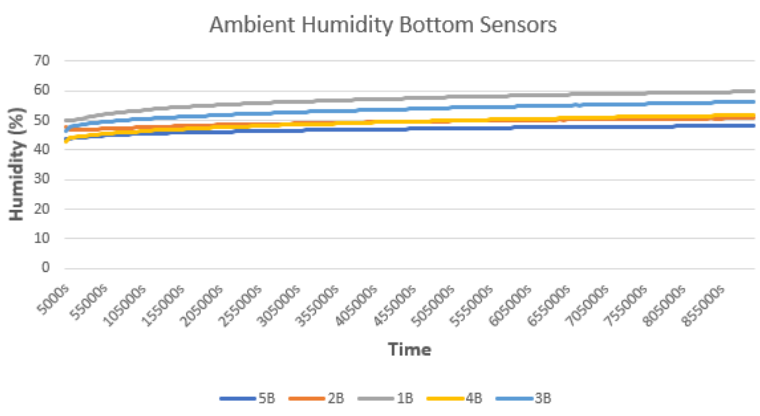


Figure 23: Ambient Low Humidity Readings

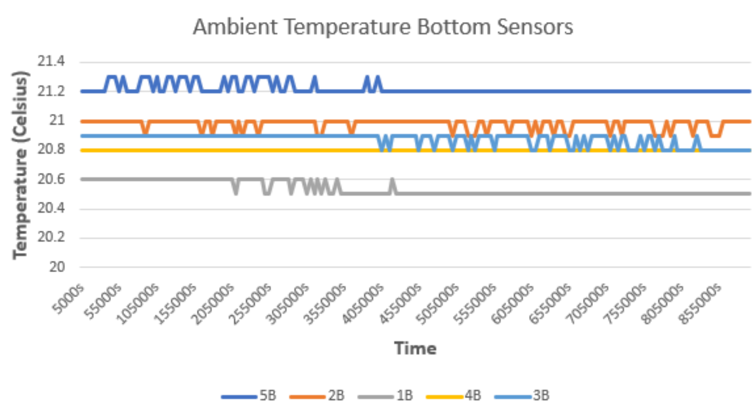


Figure 24: Ambient Low Temperature Readings

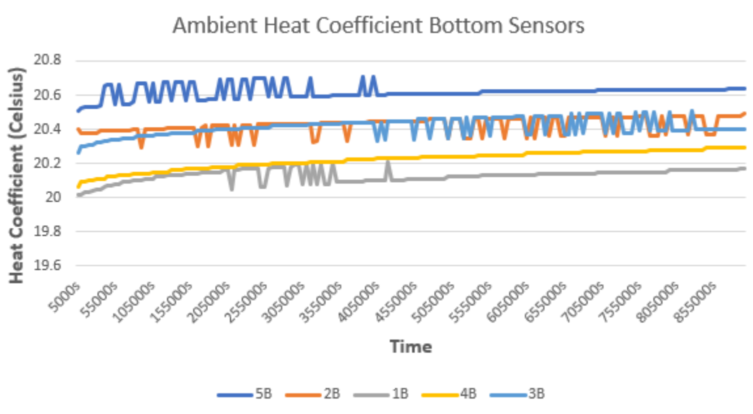


Figure 25: Ambient Low Heat Coefficient Results

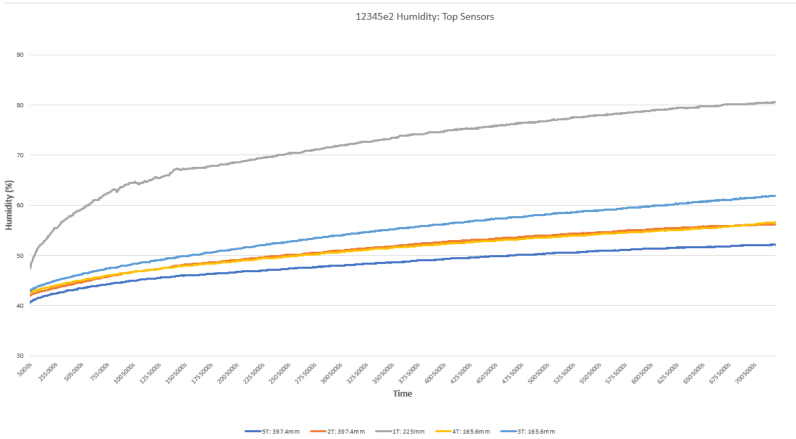


Figure 26: 12345e2 Top Humidity Readings

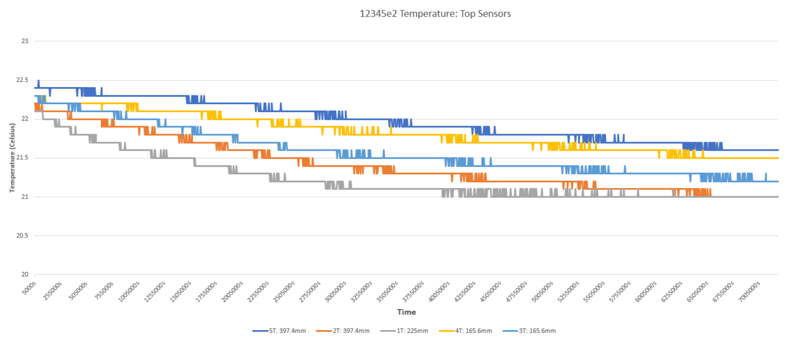


Figure 27: 12345e2 Top Temperature Readings

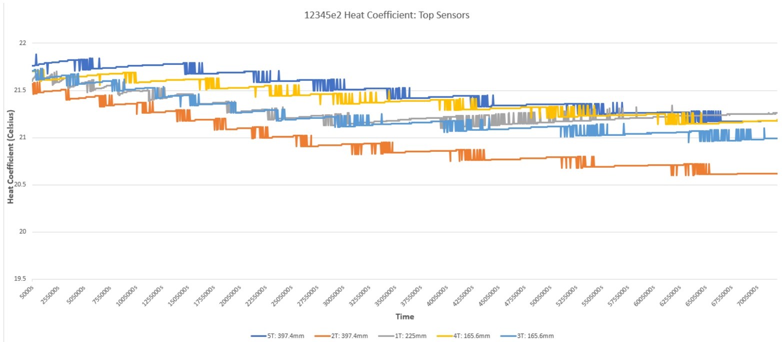


Figure 28: 12345e2 Top Heat Coefficient Results

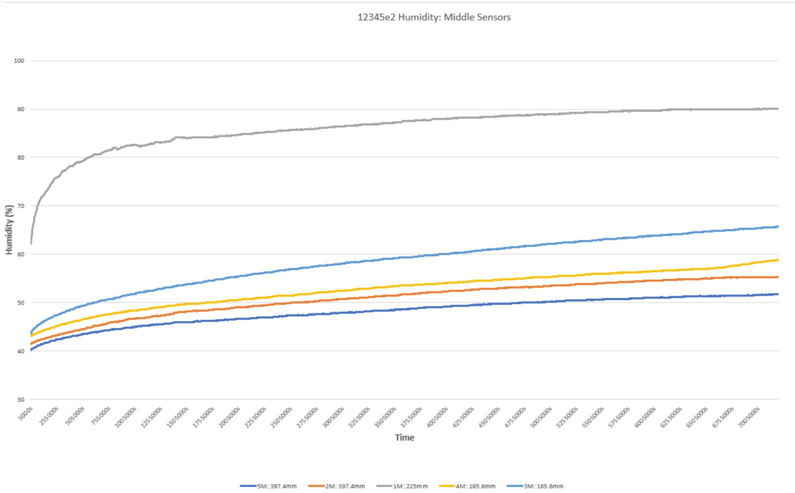


Figure 29: 12345e2 Middle Humidity Readings

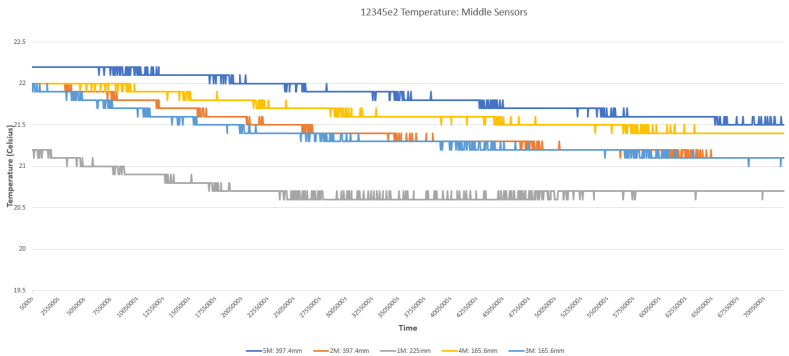


Figure 30: 12345e2 Middle Temperature Readings

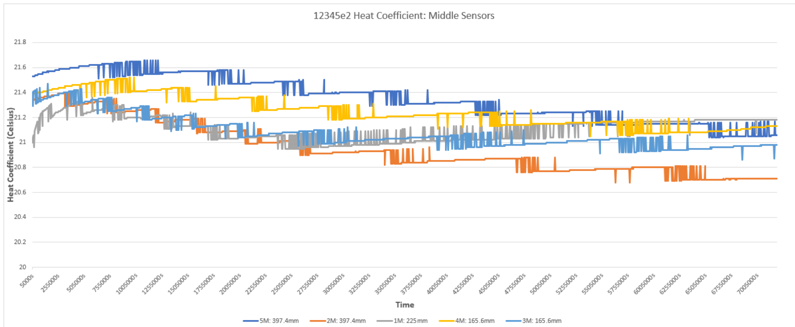


Figure 31: 12345e2 Middle Heat Coefficient Results

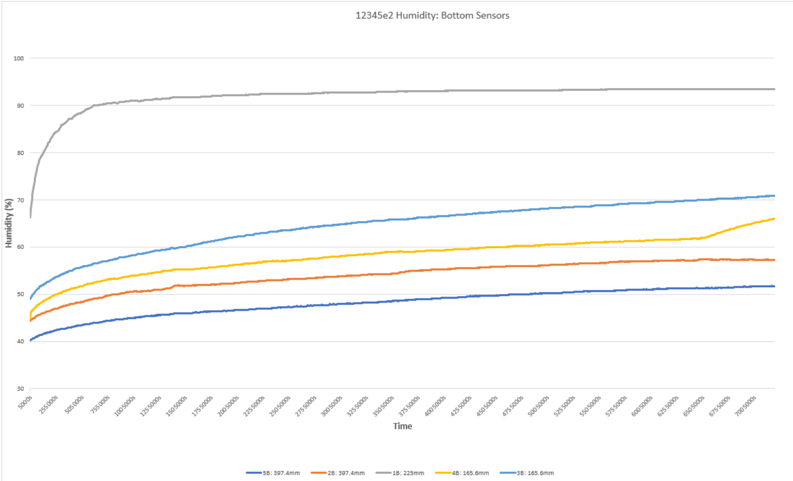


Figure 32: 12345e2 Low Humidity Readings

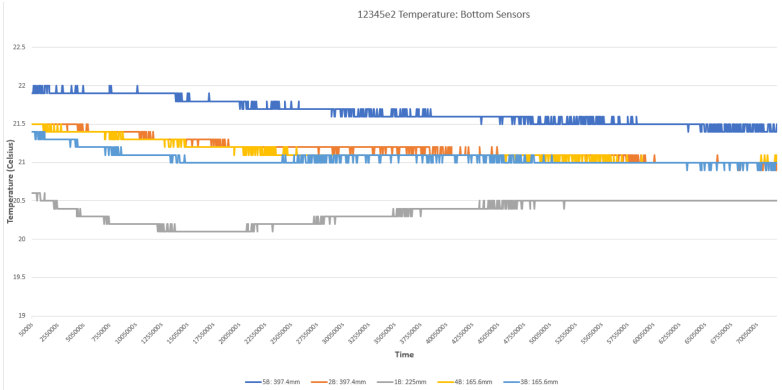


Figure 33: 12345e2 Low Temperature Readings

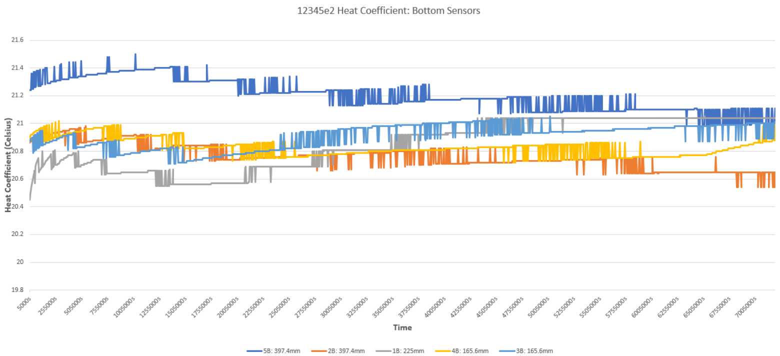


Figure 34: 12345e2 Low Heat Coefficient Results

## 6.2 Modelling

With the formulated hypotheses presented in Section 4.2 validated and data obtained to answer the research questions put forward in Section 4.3, it is now possible to use the collected data to formulate a model of the behaviour of moisture within a block of flat roof insulation. This model will be based on the collected data, and can be used in a future system which monitors humidity within an insulation system to extrapolate from readings, and estimate the distance of a leak from a sensor, the time that has elapsed since the leak became present. Through combining multiple sensor extrapolations, it should also become possible to estimate the exact location, through triangulating the combinations of distances available, given the validation of hypothesis 5 in Section 4.2, wherein moisture disperses at a constant rate from a leakage entry point.

### 6.2.1 Regression Analysis

In order to construct a suitable model of the thermal characteristics of roofing insulation and the behaviours of moisture within it, as well as the effects of moisture within an insulation block, regression analysis was carried out on the data collected in Section 6. As hypothesized in Section 4.2, moisture within an insulation block tends towards an unknown saturation point. In order to model this behaviour, the model is based on a simple lograithmic linear regression. The collected data, including regressions, for three full data sets, '12345e2', '12345e4', and '16278e5', can be seen in Appendix D.

The linear equations for the collected humidity values, and their respective horizontal and vertical distances from the moisture entry point, as well as the coefficient of determination $R^2$, are summarised in Table 6.2.2.

### 6.2.2 Linear Humidity Regression Analysis

| Horizontal Distance (mm) | Vertical Distance (mm) | Total Distance (mm) | Multiplication Coefficient | Addition Coefficient | $R^2$ |
|---|---|---|---|---|---|
| 145.1 | 30 | 148.1688564 | 1.5577 | 37.523 | 0.893 |
| 145.1 | 30 | 148.1688564 | 2.2036 | 38.855 | 0.9206 |
| 150 | 30 | 152.9705854 | 2.9804 | 34.566 | 0.9334 |
| 145.1 | 50 | 153.4731573 | 1.6356 | 36.588 | 0.9101 |
| 145.1 | 50 | 153.4731573 | 2.5267 | 37.746 | 0.9208 |
| 150 | 50 | 158.113883 | 2.8373 | 38.091 | 0.9507 |
| 145.1 | 70 | 161.1024829 | 1.9807 | 35.578 | 0.9672 |
| 145.1 | 70 | 161.1024829 | 3.3156 | 36.507 | 0.9786 |
| 150 | 70 | 165.5294536 | 3.7576 | 42.692 | 0.975 |
| 165.6 | 30 | 168.2954545 | 3.3595 | 30.03 | 0.8752 |
| 165.6 | 30 | 168.2954545 | 4.6751 | 25.262 | 0.8667 |
| 165.6 | 50 | 172.9836987 | 3.5267 | 30.683 | 0.8969 |
| 165.6 | 50 | 172.9836987 | 5.0726 | 26.562 | 0.9081 |
| 165.6 | 70 | 179.7869851 | 3.6969 | 35.003 | 0.912 |
| 165.6 | 70 | 179.7869851 | 4.9018 | 33.823 | 0.99395 |
| 225 | 30 | 226.9911893 | 7.0095 | 28.012 | 0.96 |
| 225 | 30 | 226.9911893 | 0.9427 | 42.297 | 0.9729 |
| 225 | 50 | 230.4886114 | 4.2527 | 59.48 | 0.9951 |
| 225 | 50 | 230.4886114 | 0.9568 | 40.254 | 0.9568 |
| 225 | 70 | 235.6374334 | 2.7503 | 74.698 | 0.8241 |
| 225 | 70 | 235.6374334 | 1.2093 | 36.184 | 0.9387 |
| 306.2 | 30 | 307.6661177 | 0.5907 | 42.309 | 0.964 |
| 306.2 | 50 | 310.2554431 | 1.1416 | 37.969 | 0.9523 |
| 306.2 | 70 | 314.099411 | 0.8365 | 39.889 | 0.9532 |
| 324.2 | 30 | 325.5850734 | 1.5534 | 39.593 | 0.9519 |
| 326.5 | 30 | 327.8753574 | 2.7168 | 36.826 | 0.8892 |
| 324.2 | 50 | 328.0329861 | 2.0108 | 36.403 | 0.9223 |
| 326.5 | 50 | 330.3062972 | 3.3453 | 36.195 | 0.9243 |
| 324.2 | 70 | 331.6709815 | 2.0624 | 38.392 | 0.9714 |
| 326.5 | 70 | 333.9195262 | 3.8193 | 39.366 | 0.9781 |
| 397.4 | 30 | 398.5307516 | 2.7076 | 31.334 | 0.9026 |
| 397.4 | 30 | 398.5307516 | 3.5635 | 28.941 | 0.8964 |
| 397.4 | 30 | 398.5307516 | 0.4484 | 43.653 | 0.8955 |
| 397.4 | 50 | 400.5330948 | 2.6411 | 31.619 | 0.9203 |
| 397.4 | 50 | 400.5330948 | 3.4094 | 29.551 | 0.9169 |
| 397.4 | 50 | 400.5330948 | 0.708 | 40.493 | 0.9523 |
| 397.4 | 70 | 403.5179798 | 2.6411 | 31.619 | 0.9203 |
| 397.4 | 70 | 403.5179798 | 2.9985 | 35.14 | 0.9459 |
| 397.4 | 70 | 403.5179798 | 0.7635 | 41.482 | 0.9306 |
| 493.1 | 30 | 494.0117509 | 0.3385 | 41.106 | 0.6164 |
| 493.1 | 50 | 495.6285 | 0.5743 | 39.695 | 0.7821 |
| 493.1 | 70 | 498.0437832 | 0.685 | 39.898 | 0.8152 |
| 591 | 30 | 591.7609315 | -0.145 | 43.793 | 0.1999 |
| 591 | 50 | 593.111288 | 0.1294 | 43.82 | 0.1777 |
| 591 | 70 | 595.131078 | 0.302 | 43.553 | 0.46 |

### 6.2.3 Humidity Model

Based on the collected data and regression analysis presented in 6.2.2 and D, an aggregated model was constructed representing the humidity behaviour within a block of flat roof insulation. This was based on all data with a determination coefficient higher than 0.85. The addition coefficient is based on the linear relationship between starting humidity and crossing point on the y-axis, and can be approximated by $\frac{w}{1.2381}$.

The multiplication coefficient was identified as having several key behaviours. Having confirmed the key hypothesis that rate of dispersal of moisture within a block is uniform, the two factors which contribute to the rate of dispersal at a given point in a block are the starting humidity, and the distance from a leakage entry point. As such, the model for the multiplication coefficient is approximated at $2.4672 \times 2^{\frac{m-100}{200}} \times w$.

As can be seen in Table 6.2.2, readings at a distance of over 500mm significantly decrease in determination coefficient and reliability, so it is expected this model will not be sufficient for distances of over 500mm.

Thus, the proposed model for humidity within a flat insulation block when moisture is present at a given entry point distance $m$ from the sensor taking measurements is as follows:

$$h = 2.4672w \times 2^{\frac{m-100}{200}} \log_2(x) + \frac{w}{1.2381}$$

Where:

- $h$ = Humidity (%)
- $x$ = Time (ms)
- $w$ = Ambient/Beginning Humidity (%)
- $m$ = Distance between sensor and entry point (hypotenuse of horizontal & vertical distance) (mm)

Without the presence of a leakage, insulation block humidity should remain constant, and approximate:

$$h = w$$

A visualisation of this model at distances from an entry point 100mm, 300mm and 500mm, and at starting humidities 30% (green), 35% (purple), 40% (black), 50% (red), and 75% (blue) can be seen in Figure 35. For each humidity, the lowest humidity is at 500mm, with the highest at 100mm. From this, it is possible to see how the rate of increase changes more greatly at closer distances to the entry point.

In figs. 36 to 38 the comparison between the real collected data for sensor grouping '12345' and entry point 'e2' and the model's projected humidity values is displayed; given the information on ambient humidity for each sensor placing, and distance from the leakage entry point. The red lines correspond to the real collected data, and the blue lines correspond to the model's predicted values. In a comparison for all values within this data set, the model was accurate to within 8% on average, with the sensors at placement '1' accounting for 37.5% of all individual error. This can be partially explained by the likelihood that some moisture came into direct contact with the sensors, leading to some errant readings, explaining the slightly higher inaccuracy.

When the other 2 data sets were analysed in a corresponding way, it was found that the overall accuracy for the sensor grouping '12345' placement 'e4' was accurate to within 1%, while the accuracy for sensor grouping '16278' and placement 'e5' was to within 12%. These findings also align with the earlier suggestion that this model holds to sensor placings of up to 500mm; the sensor placings in this grouping of 500mm or over total distance account for 4% of individual error.

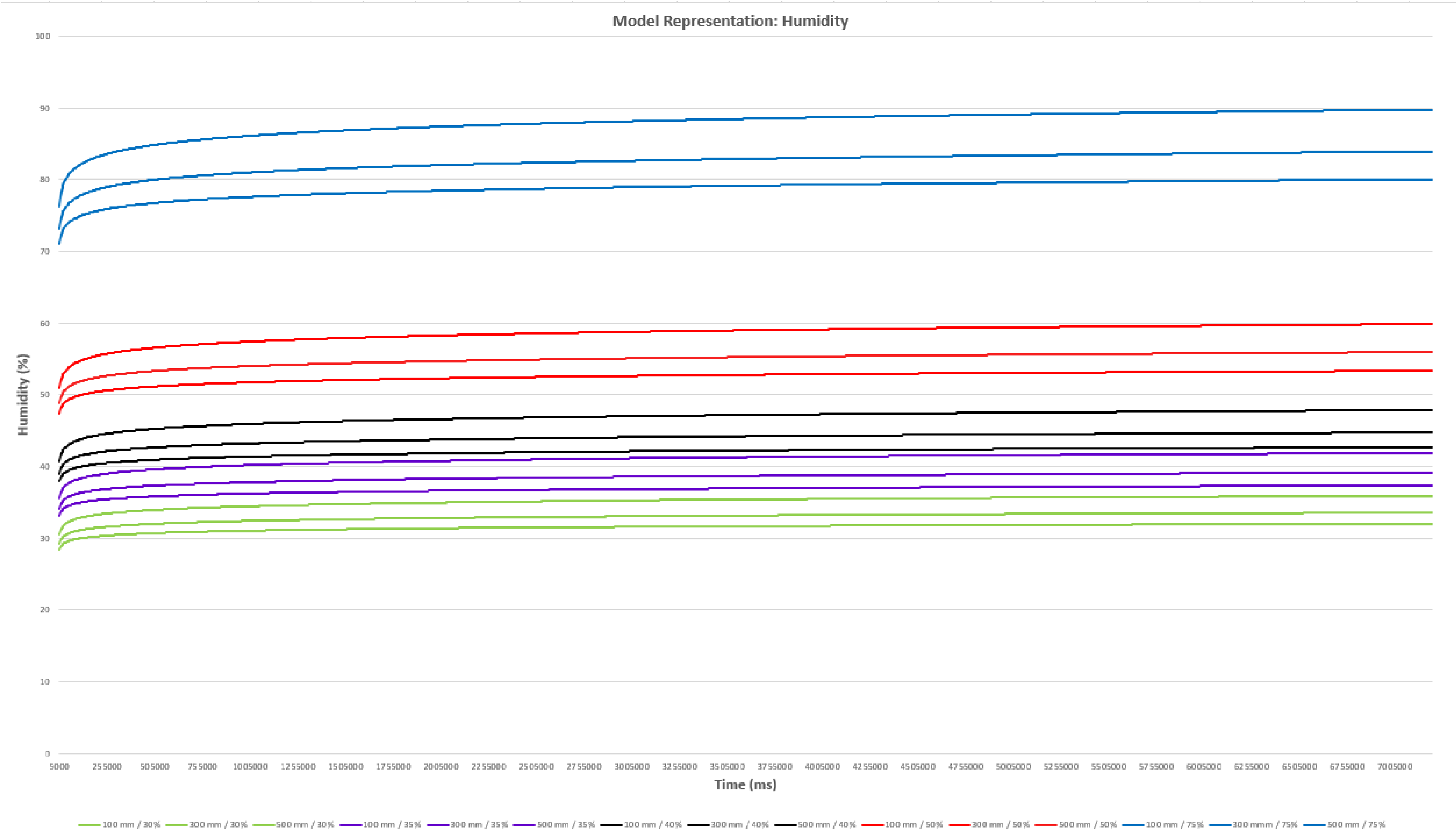


Figure 35: Humidity Model Visualisation

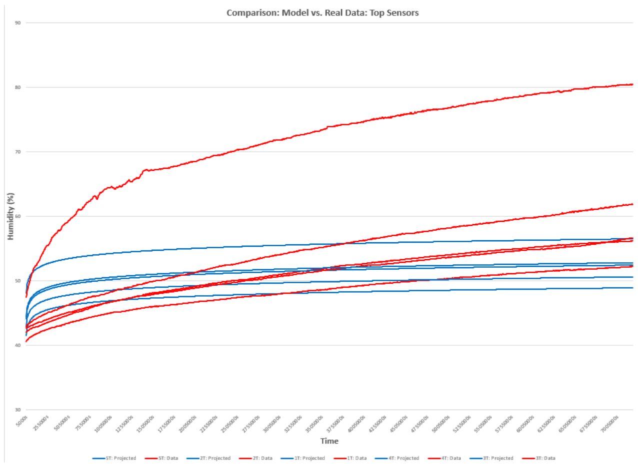


Figure 36: 12345e2 Comparison: Model to Data High Sensors

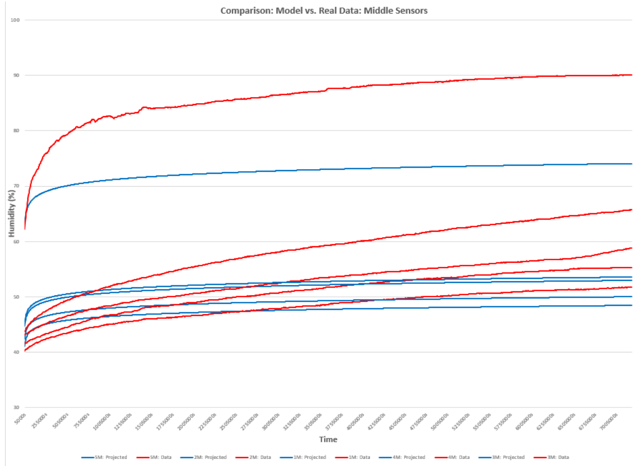


Figure 37: 12345e2 Comparison: Model to Data Middle Sensors

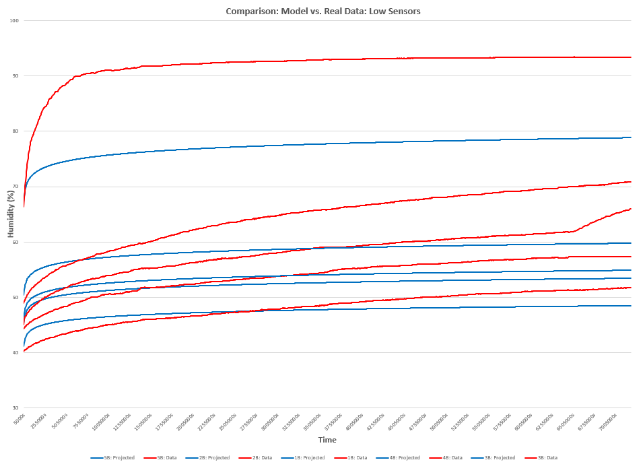


Figure 38: 12345e2 Comparison: Model to Data Low Sensors

# 7 Conclusions, Discussion & Future Works

## 7.1 Discussion

In order to assess the success of this research, it is important to discuss the results with relation to the initial Research Questions posed in Section 4.3.

1) How suitable are temperature and humidity sensors for leakage detection? Findings of this paper suggest that humidity in particular is suitable for leakage detection. Temperature is too susceptible to external changes in temperature affecting its behaviours, so it can be considered insufficient.

– What is the temperature and humidity distribution model throughout a block of insulation? The humidity distribution model is shown in Section 6.2.3. The collected temperature data was insufficient to generate a temperature distribution model.

– How does the presence of moisture within an insulation block impact the temperature and humidity distribution model? This paper's findings suggest that without the presence of moisture, humidity and temperature readings remain constant, regulating around the initial ambient temperature. When moisture is present, humidity and temperature change in relation to time and distance from the moisture entry point.

2) If temperature and humidity sensors are suitable, where should these sensors be located? These sensors should ideally be placed at several depths within the insulation blocks, spaced 500mm away from one another.

– How does distance from moisture affect sensor readings? Past distances of approximately 500mm, the accuracy of the model diminishes. However, further tests over a longer period of time may show increased accuracy with a prolonged period of moisture dissipation.

– How does depth within a block affect sensor readings? Readings correlate with the hypotenuse calculated from both horizontal and vertical distance. Thus, several depths of sensors at each point can allow for better accuracy of leak point calculation.

How suitable is this model for a future leakage detection system? This model appears to be suitable for a rudimentary leakage detection system.

– Within the model, how should sensors be placed in order to eliminate areas of no coverage? In order to eliminate areas of no coverage and to maximise the model's effectiveness, sensors should be placed in a grid pattern, with 500mm spacing between each set of 3 sensors.

– What future works and improvements are necessary for such a system? An algorithm which can calculate the meeting point of 4 models is necessary in order for this model to be usable as a leakage detection system, as this model currently only be used to calculate the hypotenuse distance from one sensor set. Additionally, it may be possible to refine this model further using a variable logarithmic base rather than a set base of 2.

## 7.2 Conclusion, Proposed System & Future Works

This research has established several trends between the presence of moisture within a block of roofing insulation and the thermal characteristics of that block, most notably an uptick in humidity levels, correlating with a downward trend in ambient temperature within the insulation block and thus the heat coefficient. However, while clear mathematical correlation between the humidity levels and the presence of moisture inserted through a single point in a block, as if to model a leak, has been observed, the scope of experiments undertaken have not fully shown the range of that correlation.

As demonstrated in Section 6.1, the internal humidity of a block of roof insulation with regards to the block's moisture content can be approximately modeled upon a logarithmic curve, with increase slowing as the block reaches an unknown saturation point. A more precise modeling of insulation behaviour could be obtained in future works through experiments testing one grouping of sensor placements over a significantly increased time duration, in order to ascertain how behaviour changes, if at all, when this saturation point is fully realised. However, as this paper aimed to propose a form of early leakage detection, knowledge of this behaviour may not be required, and the logarithmic representation is suitable. These experiments centring around extended duration should also be used to ascertain if extended duration increases the accuracy of further placed sensors, and whether such an extended duration of time leads to any part of the block becoming overly saturated.

Additionally, this research aimed to establish a model which demonstrated the characteristics of change in temperature when moisture is present within an insulation block. However, through the discrepancies introduced by external changes in temperature, the data collected was insufficient, and a model was not produced. Such a model could be produced through continued experimentation, which also measures external temperatures as well as internal temperatures. Temperature behaviours can be influenced by a wide variety of external factors, most notably sunlight. Further works investigating the effects of solar rays and outdoor conditions on the temperature model may be required in order to investigate further.

As such, this research proposes using temperature and humidity sensors to actively monitor humidity readings within a set of blocks of insulation, through a grid of sensors (at 3 depths of 30mm, 50mm and 70mm) spaced 500mm apart. Sensors maintaining ambient temperature and humidity readings can be assumed to not be within range of a leakage entry point. Conversely, any given leakage detection point should be detected by 4 surrounding sensor sets; through comparisons to the modelled behaviour discussed in this paper and the ambient temperature, triangulation of the leakage location point between these 4 sensors and calculation of the time a leakage has been present should be possible. The principle area of future works thus centres around the development of the proposed model into a leakage detection system. An algorithm which can approximate a meeting point between 4 different modeled points is required in order for this model to be applicable as a leakage detection and location system for a sensor network within a set of blocks of insulation. With this, the model proposed in this paper should suffice as a form of flat roof leakage detection.

## References


[1] D. P. Coffelt, C. T. Hendrickson "Life-Cycle Costs of Commerical Roof Systems" *Journal of Architectural Engineering*, Vol. 16, Issue 1

[2] insulationsuperstore.co.uk: "Insulating a Flat Roof", [https://www.insulationsuperstore.co.uk/pages/insulating-a-flat-roof.html], Accessed 17/01/2020.

[3] wbdg.org: "Integrity Testing for Roofing and Waterproofing Membranes", [https://www.wbdg.org/resources/integrity-testing-roofing-and-waterproofing-membranes], Accessed 17/01/2020.

[4] nrca.net: "Flood Test", [http://www.nrca.net/Technical/Search/GlossaryDetails/308/Flood-test], Accessed 17/01/2020.

[5] constructionspecifier.com: "Everything Leaks: Testing roofs to ensure watertightness at the outset", [https://www.constructionspecifier.com/everything-leaks-testing-roofs-to-ensure-watertightness-at-the-outset/], Accessed 17/01/2020.

[6] K. Roberts "The Electrical Earth Leakage Technique for Locating Holes in Roof Membranes", *Proceedings of the Fourth International Symposium on Roofing Technology*, 1997.

[7] D. Vokey "A Method to Detect and Locate Roof Leaks Using Conductive Tapes", *Moisture Measurement*, Session WB8-2.

[8] atlanticleak.com "High Voltage Leak Detection", [https://atlanticleak.com/high-voltage-leak-detection/], Accessed 17/01/2020.

[9] S. Wood "Non-Invasive Roof Leak Detection Using Infrared Thermography", *InfraMation 2004 Proceedings*, 2004.

[10] infraredimagingservices.com "Infrared roof Moisture Surveys" [http://www.infraredimagingservices.com/roof-scan-ir], Accessed 17/01/2020.

[11] roofingcontractor.com "Technical Details: How to Properly Determine Moisture Content in Roof Systems" [https://www.roofingcontractor.com/articles/83510-technical-details-how-to-properly-determine-moisture-content-in-roof-systems], Accessed 17/01/2020.

[12] detecsystems.com "PermaScan", [https://detecsystems.com/leak-detection-system/]

[13] D. Vokey "Moisture Monitoring System for Buildings", Patent #US7768412B2, [https://patents.google.com/patent/US7768412]

[14] A. DeRouin, S. Terierweiler, B. Pereles, B. Lippi, K. Ghee Ong "A Low Cost, Wireless Embedded Sensor for Moisture Monitoring in Hard-to-Access Places", *Sensor Letters*, Vol. 11, Number 9, pp. 1573-1578, 2013.

[15] buildingscience.com "Guide to Insulating Sheathing" [https://www.buildingscience.com/sites/default/files/migrate/pdf/GM_Guide_Insulating_Sheathing.pdf], Accessed 19/01/2020.

[16] adafruit.com "AM2302 (wured DHT22) temperature-humidity sensor" [https://www.adafruit.com/product/393], Accessed 17/01/2020.

# 8 Appendices

## A Arduino Microcontroller Code

```
#include "DHT.h"

//Sensor Input Pins
#define DHTPIN1 2
#define DHTPIN2 3
#define DHTPIN3 4
#define DHTPIN4 5
#define DHTPIN5 6
#define DHTPIN6 7
#define DHTPIN7 8
#define DHTPIN8 9
#define DHTPIN9 10
#define DHTPIN10 11
#define DHTPIN11 12
#define DHTPIN12 13
#define DHTPIN13 14
#define DHTPIN14 15
#define DHTPIN15 16

//Sensor Type
#define DHTTYPE DHT22

// Initialize DHT sensors.
DHT dht1(DHTPIN1, DHTTYPE);
DHT dht2(DHTPIN2, DHTTYPE);
DHT dht3(DHTPIN3, DHTTYPE);
DHT dht4(DHTPIN4, DHTTYPE);
DHT dht5(DHTPIN5, DHTTYPE);
DHT dht6(DHTPIN6, DHTTYPE);
DHT dht7(DHTPIN7, DHTTYPE);
DHT dht8(DHTPIN8, DHTTYPE);
DHT dht9(DHTPIN9, DHTTYPE);
DHT dht10(DHTPIN10, DHTTYPE);
DHT dht11(DHTPIN11, DHTTYPE);
DHT dht12(DHTPIN12, DHTTYPE);
DHT dht13(DHTPIN13, DHTTYPE);
DHT dht14(DHTPIN14, DHTTYPE);
DHT dht15(DHTPIN15, DHTTYPE);

//Float arrays to store data
float h[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float hPrevious[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float hDifference[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float t[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float tPrevious[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float tDifference[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float hic[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

//Timing variables
unsigned long previousMillis = 0, currentMillis;
const long interval = 5000;

void setup() {
  Serial.begin(19200);
  Serial.println(F("Program Beginning"));

  //Begin Sensors
  dht1.begin();
  dht2.begin();
  dht3.begin();
  dht4.begin();
  dht5.begin();
  dht6.begin();
  dht7.begin();
  dht8.begin();
  dht9.begin();
  dht10.begin();
  dht11.begin();
  dht12.begin();
  dht13.begin();
  dht14.begin();
  dht15.begin();
}

void loop() {

  //time since program elapsed
```

```
  currentMillis = millis();

  //if interval had elapsed
  if(currentMillis - previousMillis >= interval){

    previousMillis = currentMillis;

    Serial.print(previousMillis);
    Serial.println(F("ms"));

    for(int i=1; i<16 ; i++){
      readSensors(i);
      if (isnan(h[i]) || isnan(t[i])){
        Serial.print(F("Failed to read from DHT sensor "));
        Serial.println(i);
      }else{
        printResults(h[i],hDifference[i],t[i],tDifference[i],hic[i]);
        hPrevious[i]=h[i];
        tPrevious[i]=t[i];
      }
    }
  }
}

void readSensors(int i){
  if(i==1){
    h[i] = dht1.readHumidity();
    t[i] = dht1.readTemperature();
    hic[i] = dht1.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("2T "));
  }else if(i==2){
    h[i] = dht2.readHumidity();
    t[i] = dht2.readTemperature();
    hic[i] = dht2.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("2M "));
  }else if(i==3){
    h[i] = dht3.readHumidity();
    t[i] = dht3.readTemperature();
    hic[i] = dht3.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("2B "));
  }else if(i==4){
    h[i] = dht4.readHumidity();
    t[i] = dht4.readTemperature();
    hic[i] = dht4.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("5T "));
  }else if(i==5){
    h[i] = dht5.readHumidity();
    t[i] = dht5.readTemperature();
    hic[i] = dht5.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("5M "));
  }else if(i==6){
    h[i] = dht6.readHumidity();
    t[i] = dht6.readTemperature();
    hic[i] = dht6.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("5B "));
  }else if(i==7){
    h[i] = dht7.readHumidity();
    t[i] = dht7.readTemperature();
    hic[i] = dht7.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("1T "));
  }else if(i==8){
    h[i] = dht8.readHumidity();
    t[i] = dht8.readTemperature();
    hic[i] = dht8.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("1M "));
  }else if(i==9){
    h[i] = dht9.readHumidity();
    t[i] = dht9.readTemperature();
    hic[i] = dht9.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("1B "));
  }else if(i==10){
    h[i] = dht10.readHumidity();
    t[i] = dht10.readTemperature();
    hic[i] = dht10.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("4T "));
  }else if(i==11){
    h[i] = dht11.readHumidity();
    t[i] = dht11.readTemperature();
    hic[i] = dht11.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("4M "));
  }else if(i==12){
    h[i] = dht12.readHumidity();
    t[i] = dht12.readTemperature();
```

```
    hic[i] = dht12.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("4B "));
  }else if(i==13){
    h[i] = dht13.readHumidity();
    t[i] = dht13.readTemperature();
    hic[i] = dht13.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("3T "));
  }else if(i==14){
    h[i] = dht14.readHumidity();
    t[i] = dht14.readTemperature();
    hic[i] = dht14.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("3M "));
  }else if(i==15){
    h[i] = dht15.readHumidity();
    t[i] = dht15.readTemperature();
    hic[i] = dht15.computeHeatIndex(t[i], h[i], false);
    Serial.print(F("3B "));
  }else{
  }
  hDifference[i]=h[i]-hPrevious[i];
  tDifference[i]=t[i]-tPrevious[i];
}

void printResults(float h,float hDifference,float t,float tDifference,float hic){
  Serial.print(h);
  Serial.print(F(" "));
  Serial.print(hDifference);
  Serial.print(F(" "));
  Serial.print(t);
  Serial.print(F(" "));
  Serial.print(tDifference);
  Serial.print(F(" "));
  Serial.println(hic);
}
```

# B Python Receiver Code

```
import serial

serial_port = 'COM4'
baud_rate = 19200 #In arduino, Serial.begin(baud_rate)
write_to_file_path = "12345e2.txt"

output_file = open(write_to_file_path, "w+")
ser = serial.Serial(serial_port, baud_rate)

while True:

    line = ser.readline()
    line = line.decode("utf-8") #ser.readline returns a binary, convert to string
    print(line)
    output_file.write(line)

    if(line[0:9]=="7205000ms"):

      exit()
      print("Program Ended")
```

# C Python File Splitter

```
pathway = '12345e2.txt'

timepath = 'timestamps.txt'
timeout = open(timepath, "w+")

hum2tpath = 'hum2t.txt'
hum2tout = open(hum2tpath,"w+")
hum2tdeltapath = 'hum2tdelta.txt'
hum2tdeltaout = open(hum2tdeltapath, "w+")
temp2tpath = 'temp2t.txt'
temp2tout = open(temp2tpath, "w+")
temp2tdeltapath = 'temp2tdelta.txt'
temp2tdeltaout = open(temp2tdeltapath, "w+")
temp2tcpath = 'tempc2t.txt'
temp2tcout = open(temp2tcpath, "w+")

hum2mpath = 'hum2m.txt'
hum2mout = open(hum2mpath,"w+")
hum2mdeltapath = 'hum2mdelta.txt'
hum2mdeltaout = open(hum2mdeltapath, "w+")
temp2mpath = 'temp2m.txt'
temp2mout = open(temp2mpath, "w+")
temp2mdeltapath = 'temp2mdelta.txt'
temp2mdeltaout = open(temp2mdeltapath, "w+")
temp2mcpath = 'tempc2m.txt'
temp2mcout = open(temp2mcpath, "w+")

hum2bpath = 'hum2b.txt'
hum2bout = open(hum2bpath,"w+")
hum2bdeltapath = 'hum2bdelta.txt'
hum2bdeltaout = open(hum2bdeltapath, "w+")
temp2bpath = 'temp2b.txt'
temp2bout = open(temp2bpath, "w+")
temp2bdeltapath = 'temp2bdelta.txt'
temp2bdeltaout = open(temp2bdeltapath, "w+")
temp2bcpath = 'tempc2b.txt'
temp2bcout = open(temp2bcpath, "w+")

hum5tpath = 'hum5t.txt'
hum5tout = open(hum5tpath,"w+")
hum5tdeltapath = 'hum5tdelta.txt'
hum5tdeltaout = open(hum5tdeltapath, "w+")
temp5tpath = 'temp5t.txt'
temp5tout = open(temp5tpath, "w+")
temp5tdeltapath = 'temp5tdelta.txt'
temp5tdeltaout = open(temp5tdeltapath, "w+")
temp5tcpath = 'tempc5t.txt'
temp5tcout = open(temp5tcpath, "w+")

hum5mpath = 'hum5m.txt'
hum5mout = open(hum5mpath,"w+")
hum5mdeltapath = 'hum5mdelta.txt'
hum5mdeltaout = open(hum5mdeltapath, "w+")
temp5mpath = 'temp5m.txt'
temp5mout = open(temp5mpath, "w+")
temp5mdeltapath = 'temp5mdelta.txt'
temp5mdeltaout = open(temp5mdeltapath, "w+")
temp5mcpath = 'tempc5m.txt'
temp5mcout = open(temp5mcpath, "w+")

hum5bpath = 'hum5b.txt'
hum5bout = open(hum5bpath,"w+")
hum5bdeltapath = 'hum5bdelta.txt'
hum5bdeltaout = open(hum5bdeltapath, "w+")
temp5bpath = 'temp5b.txt'
temp5bout = open(temp5bpath, "w+")
temp5bdeltapath = 'temp5bdelta.txt'
temp5bdeltaout = open(temp5bdeltapath, "w+")
temp5bcpath = 'tempc5b.txt'
temp5bcout = open(temp5bcpath, "w+")

hum1tpath = 'hum1t.txt'
hum1tout = open(hum1tpath,"w+")
hum1tdeltapath = 'hum1tdelta.txt'
hum1tdeltaout = open(hum1tdeltapath, "w+")
temp1tpath = 'temp1t.txt'
temp1tout = open(temp1tpath, "w+")
temp1tdeltapath = 'temp1tdelta.txt'
temp1tdeltaout = open(temp1tdeltapath, "w+")
temp1tcpath = 'tempc1t.txt'
temp1tcout = open(temp1tcpath, "w+")
```

```

hum1mpath = 'hum1m.txt'
hum1mout = open(hum1mpath,"w+")
hum1mdeltapath = 'hum1mdelta.txt'
hum1mdeltaout = open(hum1mdeltapath, "w+")
temp1mpath = 'temp1m.txt'
temp1mout = open(temp1mpath, "w+")
temp1mdeltapath = 'temp1mdelta.txt'
temp1mdeltaout = open(temp1mdeltapath, "w+")
temp1mcpath = 'tempc1m.txt'
temp1mcout = open(temp1mcpath, "w+")

hum1bpath = 'hum1b.txt'
hum1bout = open(hum1bpath,"w+")
hum1bdeltapath = 'hum1bdelta.txt'
hum1bdeltaout = open(hum1bdeltapath, "w+")
temp1bpath = 'temp1b.txt'
temp1bout = open(temp1bpath, "w+")
temp1bdeltapath = 'temp1bdelta.txt'
temp1bdeltaout = open(temp1bdeltapath, "w+")
temp1bcpath = 'tempc1b.txt'
temp1bcout = open(temp1bcpath, "w+")

hum4tpath = 'hum4t.txt'
hum4tout = open(hum4tpath,"w+")
hum4tdeltapath = 'hum4tdelta.txt'
hum4tdeltaout = open(hum4tdeltapath, "w+")
temp4tpath = 'temp4t.txt'
temp4tout = open(temp4tpath, "w+")
temp4tdeltapath = 'temp4tdelta.txt'
temp4tdeltaout = open(temp4tdeltapath, "w+")
temp4tcpath = 'tempc4t.txt'
temp4tcout = open(temp4tcpath, "w+")

hum4mpath = 'hum4m.txt'
hum4mout = open(hum4mpath,"w+")
hum4mdeltapath = 'hum4mdelta.txt'
hum4mdeltaout = open(hum4mdeltapath, "w+")
temp4mpath = 'temp4m.txt'
temp4mout = open(temp4mpath, "w+")
temp4mdeltapath = 'temp4mdelta.txt'
temp4mdeltaout = open(temp4mdeltapath, "w+")
temp4mcpath = 'tempc4m.txt'
temp4mcout = open(temp4mcpath, "w+")

hum4bpath = 'hum4b.txt'
hum4bout = open(hum4bpath,"w+")
hum4bdeltapath = 'hum4bdelta.txt'
hum4bdeltaout = open(hum4bdeltapath, "w+")
temp4bpath = 'temp4b.txt'
temp4bout = open(temp4bpath, "w+")
temp4bdeltapath = 'temp4bdelta.txt'
temp4bdeltaout = open(temp4bdeltapath, "w+")
temp4bcpath = 'tempc4b.txt'
temp4bcout = open(temp4bcpath, "w+")

hum3tpath = 'hum3t.txt'
hum3tout = open(hum3tpath,"w+")
hum3tdeltapath = 'hum3tdelta.txt'
hum3tdeltaout = open(hum3tdeltapath, "w+")
temp3tpath = 'temp3t.txt'
temp3tout = open(temp3tpath, "w+")
temp3tdeltapath = 'temp3tdelta.txt'
temp3tdeltaout = open(temp3tdeltapath, "w+")
temp3tcpath = 'tempc3t.txt'
temp3tcout = open(temp3tcpath, "w+")

hum3mpath = 'hum3m.txt'
hum3mout = open(hum3mpath,"w+")
hum3mdeltapath = 'hum3mdelta.txt'
hum3mdeltaout = open(hum3mdeltapath, "w+")
temp3mpath = 'temp3m.txt'
temp3mout = open(temp3mpath, "w+")
temp3mdeltapath = 'temp3mdelta.txt'
temp3mdeltaout = open(temp3mdeltapath, "w+")
temp3mcpath = 'tempc3m.txt'
temp3mcout = open(temp3mcpath, "w+")

hum3bpath = 'hum3b.txt'
hum3bout = open(hum3bpath,"w+")
hum3bdeltapath = 'hum3bdelta.txt'
hum3bdeltaout = open(hum3bdeltapath, "w+")
temp3bpath = 'temp3b.txt'
```

```
temp3bout = open(temp3bpath, "w+")
temp3bdeltapath = 'temp3bdelta.txt'
temp3bdeltaout = open(temp3bdeltapath, "w+")
temp3bcpath = 'tempc3b.txt'
temp3bcout = open(temp3bcpath, "w+")


with open(pathway, "r") as temp:

  linecnt = 0
  whole = temp.readlines()

  while whole:

    if whole[linecnt] == '\n':

      linecnt += 1

    else:

      line = whole[linecnt]
      strcnt = 0
      xcnt = 0
      xcnt1 = 0
      xcnt2 = 0
      xcnt3 = 0
      xcnt4 = 0
      xcnt5 = 0
      sensor = []

      newline = "\n"

      for x in line:

        if x == " ":

          strcnt += 1

          if strcnt == 1:

            string = (line[0:xcnt]).strip()

            if string == "2T":
              sensor = string
            if string == "2M":
              sensor = string
            if string == "2B":
              sensor = string
            if string == "5T":
              sensor = string
            if string == "5M":
              sensor = string
            if string == "5B":
              sensor = string
            if string == "1T":
              sensor = string
            if string == "1M":
              sensor = string
            if string == "1B":
              sensor = string
            if string == "4T":
              sensor = string
            if string == "4M":
              sensor = string
            if string == "4B":
              sensor = string
            if string == "3T":
              sensor = string
            if string == "3M":
              sensor = string
            if string == "3B":
              sensor = string

            xcnt1 = xcnt

          elif strcnt == 2:

            string = (line[xcnt1:xcnt]).strip()
            xcnt2 = xcnt

            if sensor == "2T":
              string = string + newline
              hum2tout.write(string)
```

```
            if sensor == "2M":
              string = string + newline
              hum2mout.write(string)
            if sensor == "2B":
              string = string + newline
              hum2bout.write(string)
            if sensor == "5T":
              string = string + newline
              hum5tout.write(string)
            if sensor == "5M":
              string = string + newline
              hum5mout.write(string)
            if sensor == "5B":
              string = string + newline
              hum5bout.write(string)
            if sensor == "1T":
              string = string + newline
              hum1tout.write(string)
            if sensor == "1M":
              string = string + newline
              hum1mout.write(string)
            if sensor == "1B":
              string = string + newline
              hum1bout.write(string)
            if sensor == "4T":
              string = string + newline
              hum4tout.write(string)
            if sensor == "4M":
              string = string + newline
              hum4mout.write(string)
            if sensor == "4B":
              string = string + newline
              hum4bout.write(string)
            if sensor == "3T":
              string = string + newline
              hum3tout.write(string)
            if sensor == "3M":
              string = string + newline
              hum3mout.write(string)
            if sensor == "3B":
              string = string + newline
              hum3bout.write(string)

          elif strcnt == 3:

            string = (line[xcnt2:xcnt]).strip()
            xcnt3 = xcnt

            if sensor == "2T":
              string = string + newline
              hum2tdeltaout.write(string)
            if sensor == "2M":
              string = string + newline
              hum2mdeltaout.write(string)
            if sensor == "2B":
              string = string + newline
              hum2bdeltaout.write(string)
            if sensor == "5T":
              string = string + newline
              hum5tdeltaout.write(string)
            if sensor == "5M":
              string = string + newline
              hum5mdeltaout.write(string)
            if sensor == "5B":
              string = string + newline
              hum5bdeltaout.write(string)
            if sensor == "1T":
              string = string + newline
              hum1tdeltaout.write(string)
            if sensor == "1M":
              string = string + newline
              hum1mdeltaout.write(string)
            if sensor == "1B":
              string = string + newline
              hum1bdeltaout.write(string)
            if sensor == "4T":
              string = string + newline
              hum4tdeltaout.write(string)
            if sensor == "4M":
              string = string + newline
              hum4mdeltaout.write(string)
            if sensor == "4B":
              string = string + newline
```

```
          hum4bdeltaout.write(string)
        if sensor == ”3T”:
          string = string + newline
          hum3tdeltaout.write(string)
        if sensor == ”3M”:
          string = string + newline
          hum3mdeltaout.write(string)
        if sensor == ”3B”:
          string = string + newline
          hum3bdeltaout.write(string)

      elif strcnt == 4:

        string = (line[xcnt3:xcnt]).strip()
        xcnt4 = xcnt

        if sensor == ”2T”:
          string = string + newline
          temp2tout.write(string)
        if sensor == ”2M”:
          string = string + newline
          temp2mout.write(string)
        if sensor == ”2B”:
          string = string + newline
          temp2bout.write(string)
        if sensor == ”5T”:
          string = string + newline
          temp5tout.write(string)
        if sensor == ”5M”:
          string = string + newline
          temp5mout.write(string)
        if sensor == ”5B”:
          string = string + newline
          temp5bout.write(string)
        if sensor == ”1T”:
          string = string + newline
          temp1tout.write(string)
        if sensor == ”1M”:
          string = string + newline
          temp1mout.write(string)
        if sensor == ”1B”:
          string = string + newline
          temp1bout.write(string)
        if sensor == ”4T”:
          string = string + newline
          temp4tout.write(string)
        if sensor == ”4M”:
          string = string + newline
          temp4mout.write(string)
        if sensor == ”4B”:
          string = string + newline
          temp4bout.write(string)
        if sensor == ”3T”:
          string = string + newline
          temp3tout.write(string)
        if sensor == ”3M”:
          string = string + newline
          temp3mout.write(string)
        if sensor == ”3B”:
          string = string + newline
          temp3bout.write(string)

      elif strcnt == 5:

        string = (line[xcnt4:xcnt]).strip()
        xcnt5 = xcnt

        if sensor == ”2T”:
          string = string + newline
          temp2tdeltaout.write(string)
        if sensor == ”2M”:
          string = string + newline
          temp2mdeltaout.write(string)
        if sensor == ”2B”:
          string = string + newline
          temp2bdeltaout.write(string)
        if sensor == ”5T”:
          string = string + newline
          temp5tdeltaout.write(string)
        if sensor == ”5M”:
          string = string + newline
          temp5mdeltaout.write(string)
        if sensor == ”5B”:
```

```
              string = string + newline
              temp5bdeltaout.write(string)
            if sensor == "1T":
              string = string + newline
              temp1tdeltaout.write(string)
            if sensor == "1M":
              string = string + newline
              temp1mdeltaout.write(string)
            if sensor == "1B":
              string = string + newline
              temp1bdeltaout.write(string)
            if sensor == "4T":
              string = string + newline
              temp4tdeltaout.write(string)
            if sensor == "4M":
              string = string + newline
              temp4mdeltaout.write(string)
            if sensor == "4B":
              string = string + newline
              temp4bdeltaout.write(string)
            if sensor == "3T":
              string = string + newline
              temp3tdeltaout.write(string)
            if sensor == "3M":
              string = string + newline
              temp3mdeltaout.write(string)
            if sensor == "3B":
              string = string + newline
              temp3bdeltaout.write(string)

        if x == '\n':
          string = (line[xcnt5:xcnt]).strip()

          if sensor == []:
            time = string
            if time != "Program Beginning":
              string = time + newline
              timeout.write(string)
          else:
            if sensor == "2T":
              string = string + newline
              temp2tcout.write(string)
            if sensor == "2M":
              string = string + newline
              temp2mcout.write(string)
            if sensor == "2B":
              string = string + newline
              temp2bcout.write(string)
            if sensor == "5T":
              string = string + newline
              temp5tcout.write(string)
            if sensor == "5M":
              string = string + newline
              temp5mcout.write(string)
            if sensor == "5B":
              string = string + newline
              temp5bcout.write(string)
            if sensor == "1T":
              string = string + newline
              temp1tcout.write(string)
            if sensor == "1M":
              string = string + newline
              temp1mcout.write(string)
            if sensor == "1B":
              string = string + newline
              temp1bcout.write(string)
            if sensor == "4T":
              string = string + newline
              temp4tcout.write(string)
            if sensor == "4M":
              string = string + newline
              temp4mcout.write(string)
            if sensor == "4B":
              string = string + newline
              temp4bcout.write(string)
            if sensor == "3T":
              string = string + newline
              temp3tcout.write(string)
            if sensor == "3M":
              string = string + newline
              temp3mcout.write(string)
            if sensor == "3B":
              string = string + newline
```

```
                temp3bcout.write(string)
        xcnt += 1
    linecnt += 1
```

# D Regression Graphs

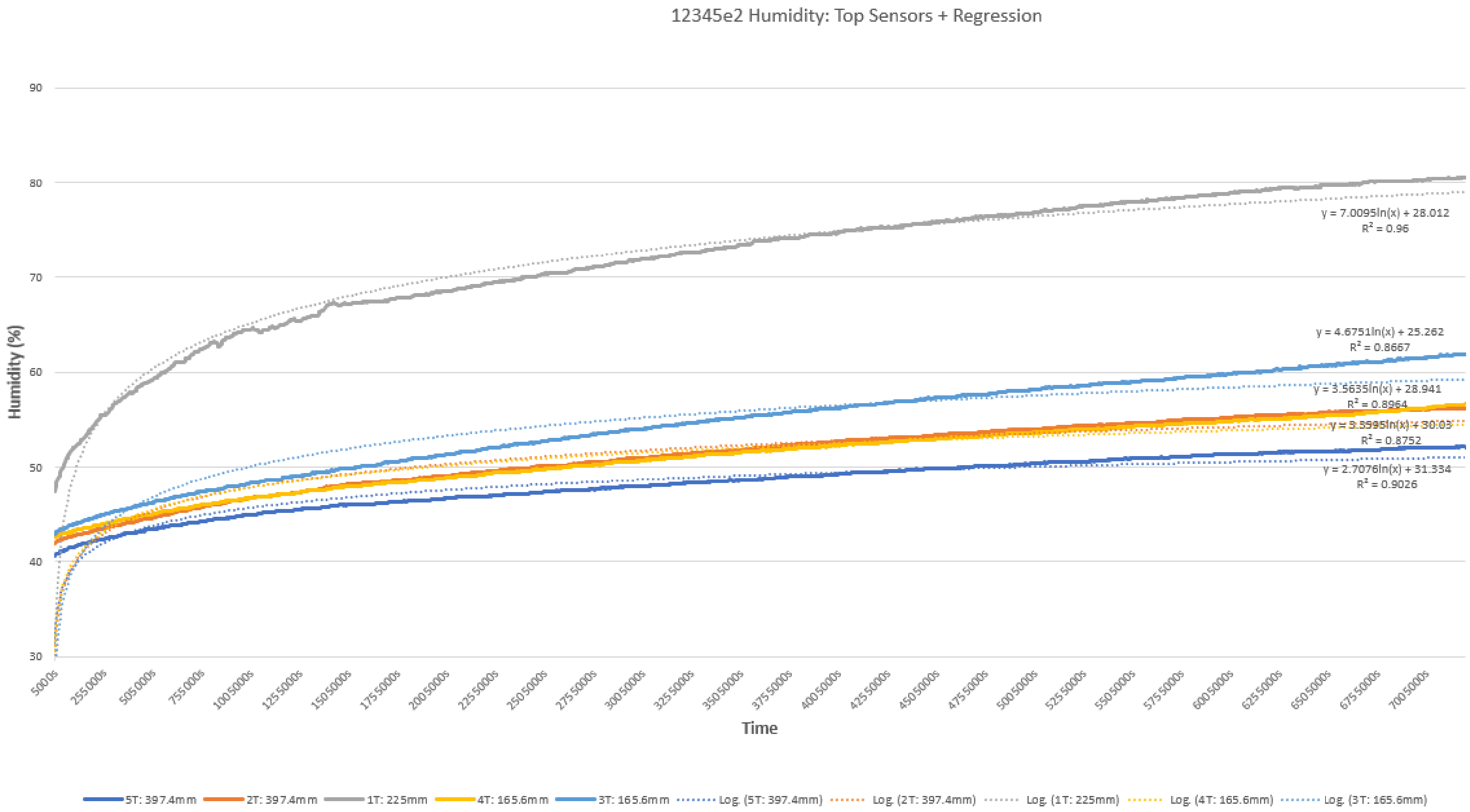


Figure 39: 12345e2 High Humidity Sensors + Regression

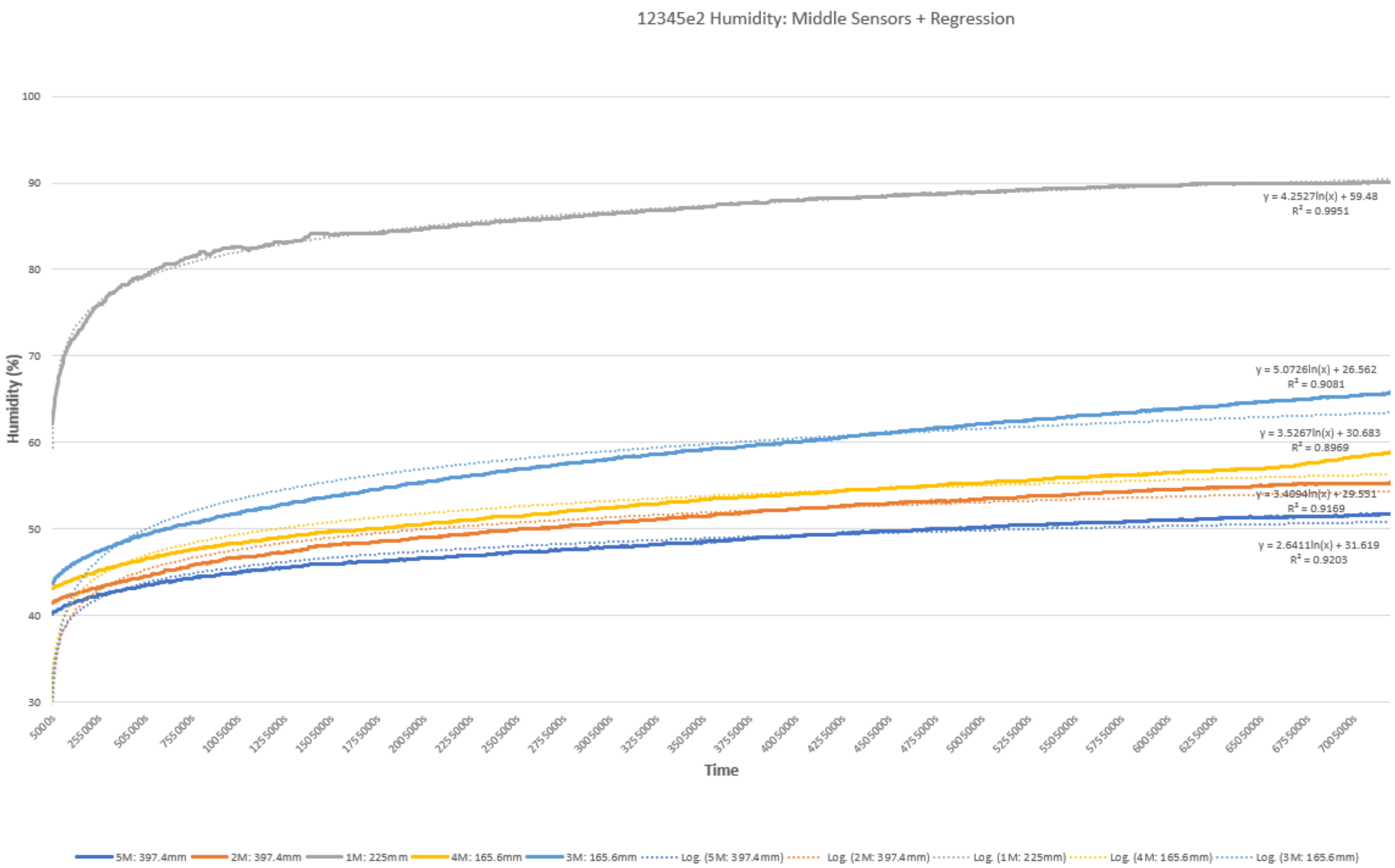


Figure 40: 12345e2 Middle Humidity Sensors + Regression

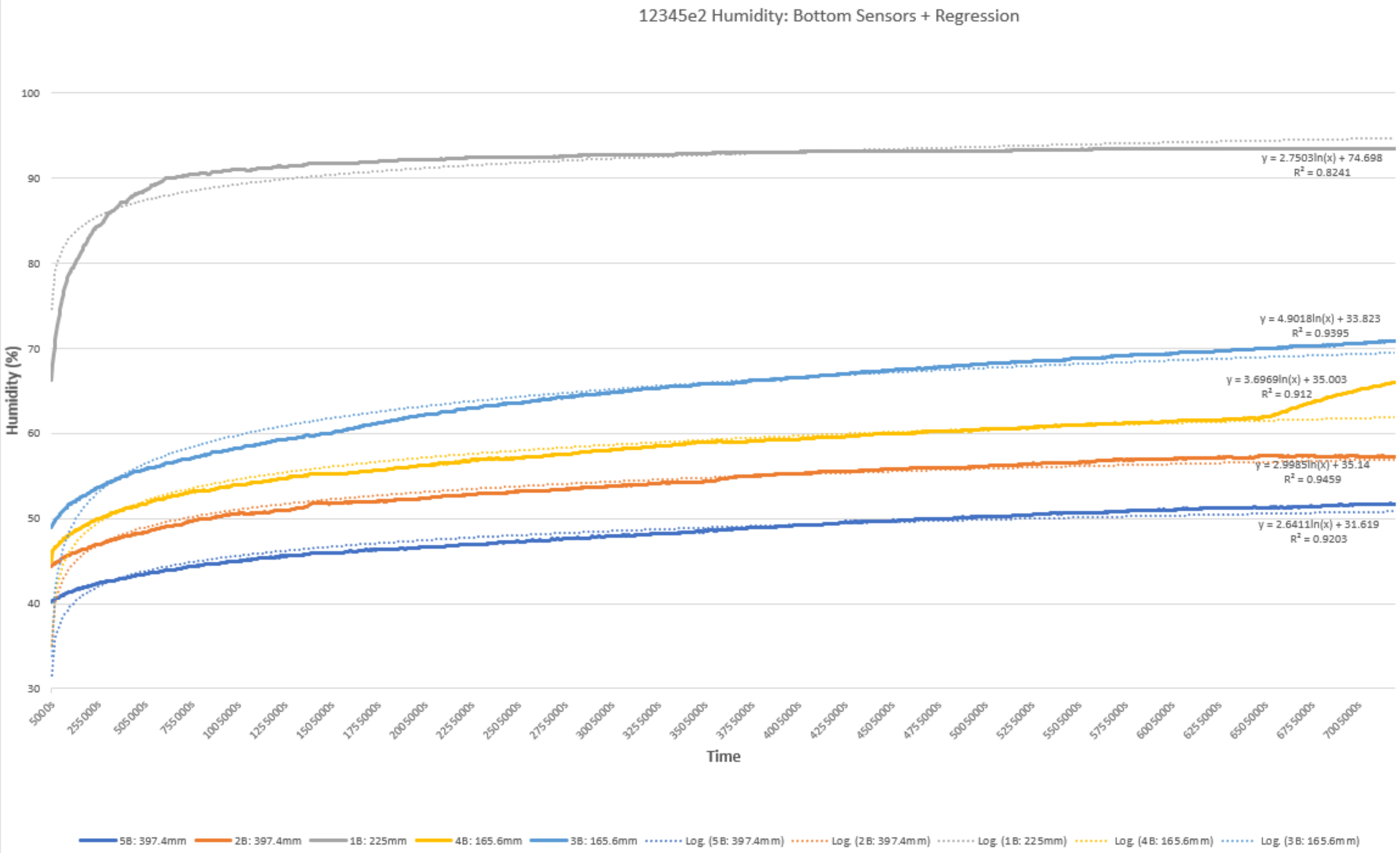


Figure 41: 12345e2 Low Humidity Sensors + Regression

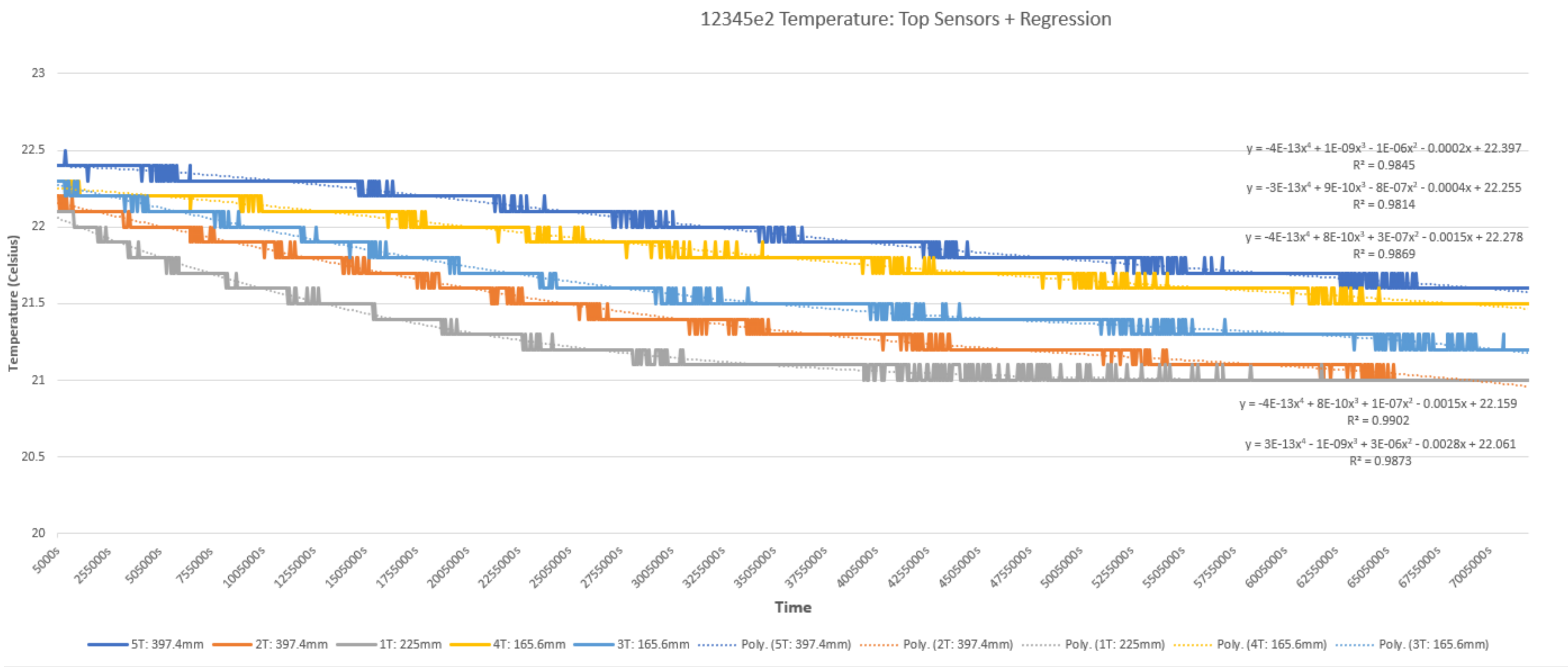


Figure 42: 12345e2 High Temperature Sensors + Regression

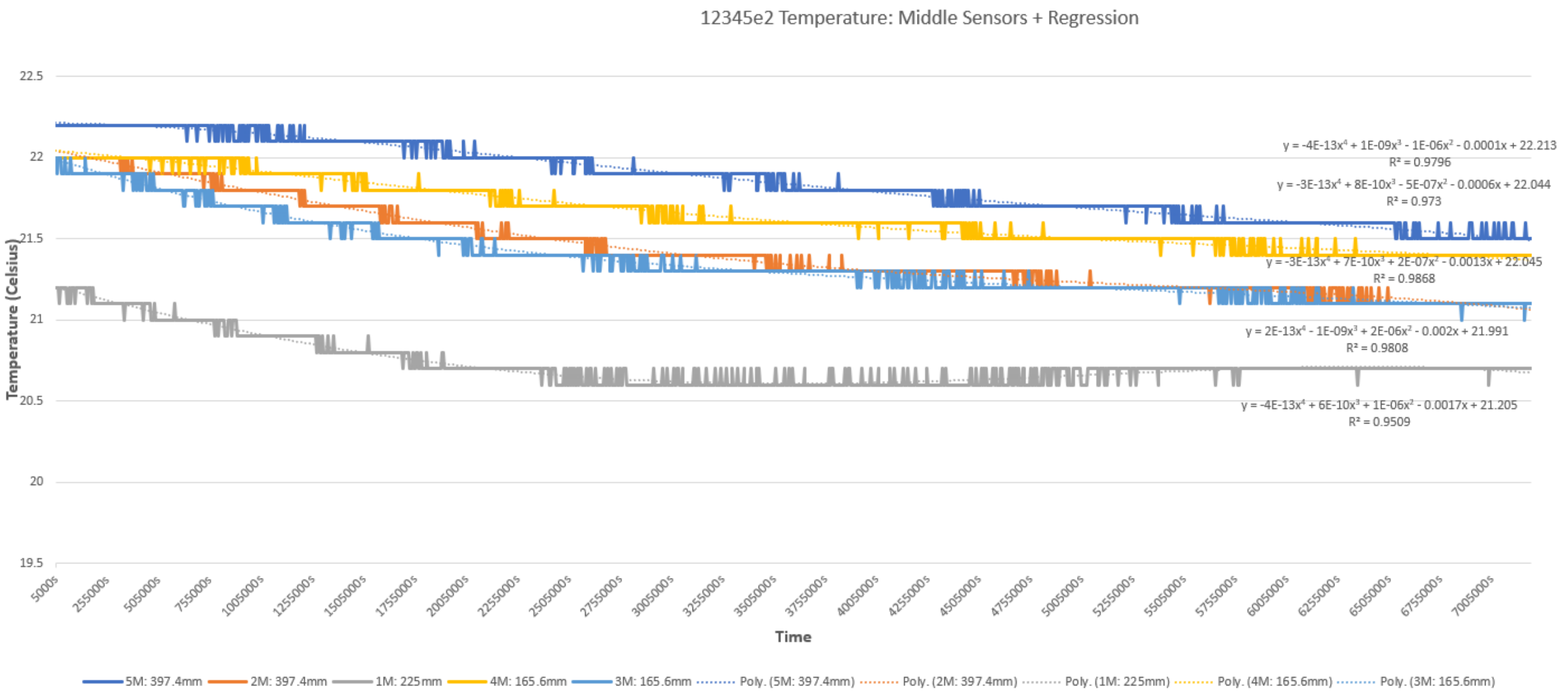


Figure 43: 12345e2 Middle Temperature Sensors + Regression

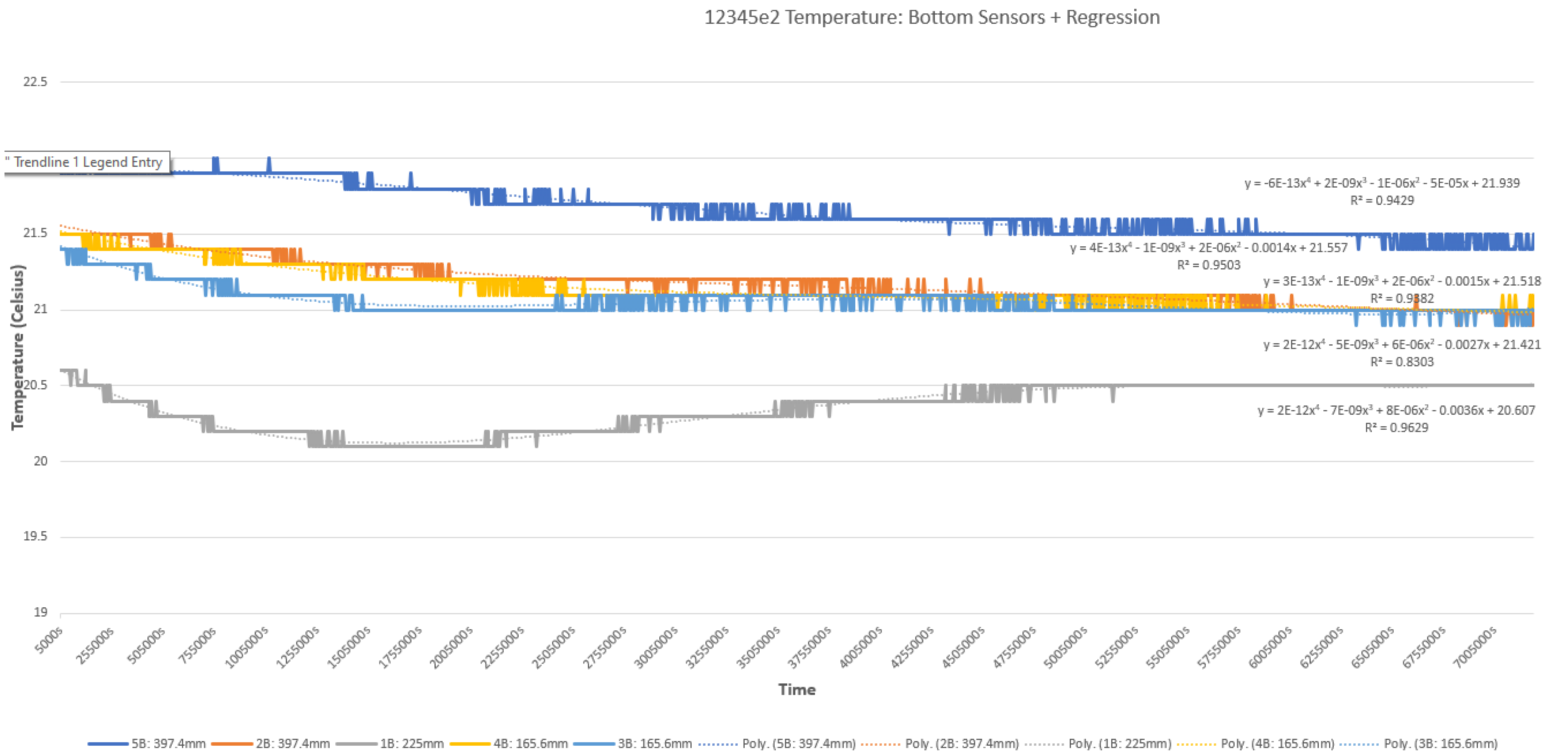


Figure 44: 12345e2 Low Temperature Sensors + Regression

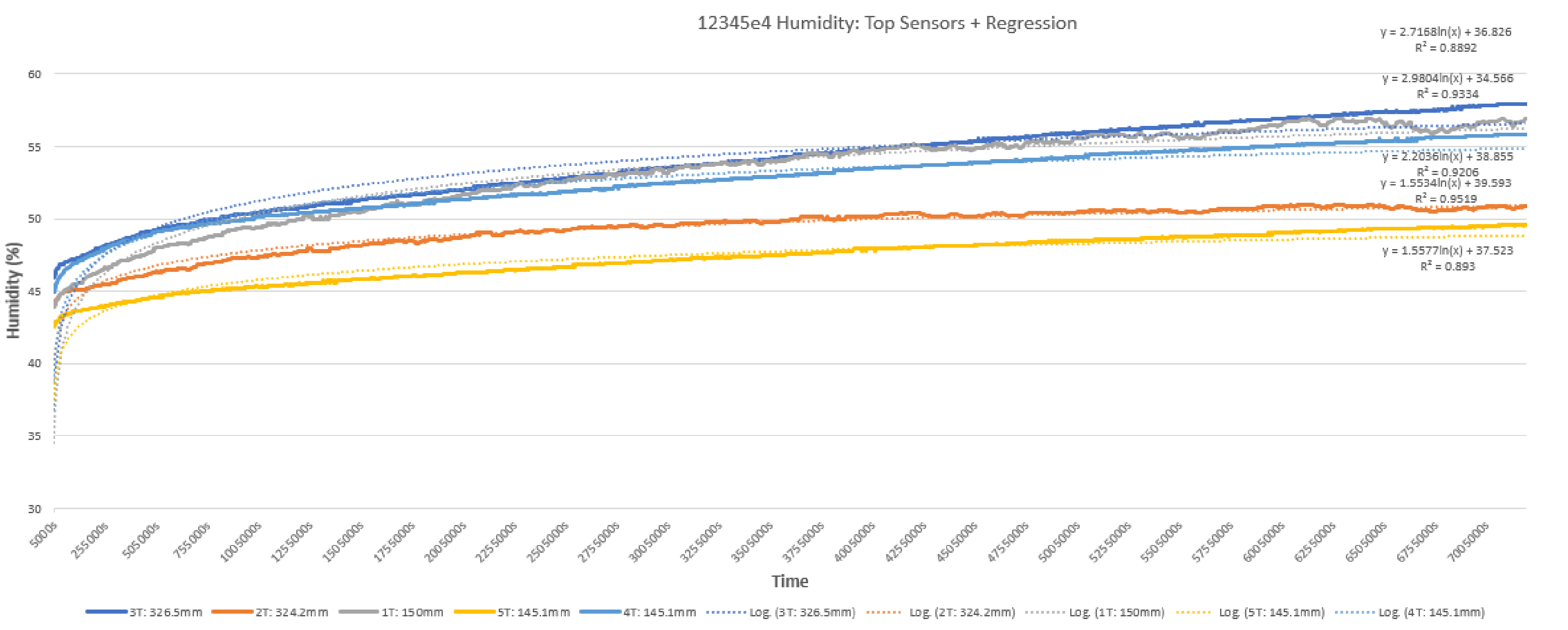


Figure 45: 12345e4 High Humidity Sensors + Regression

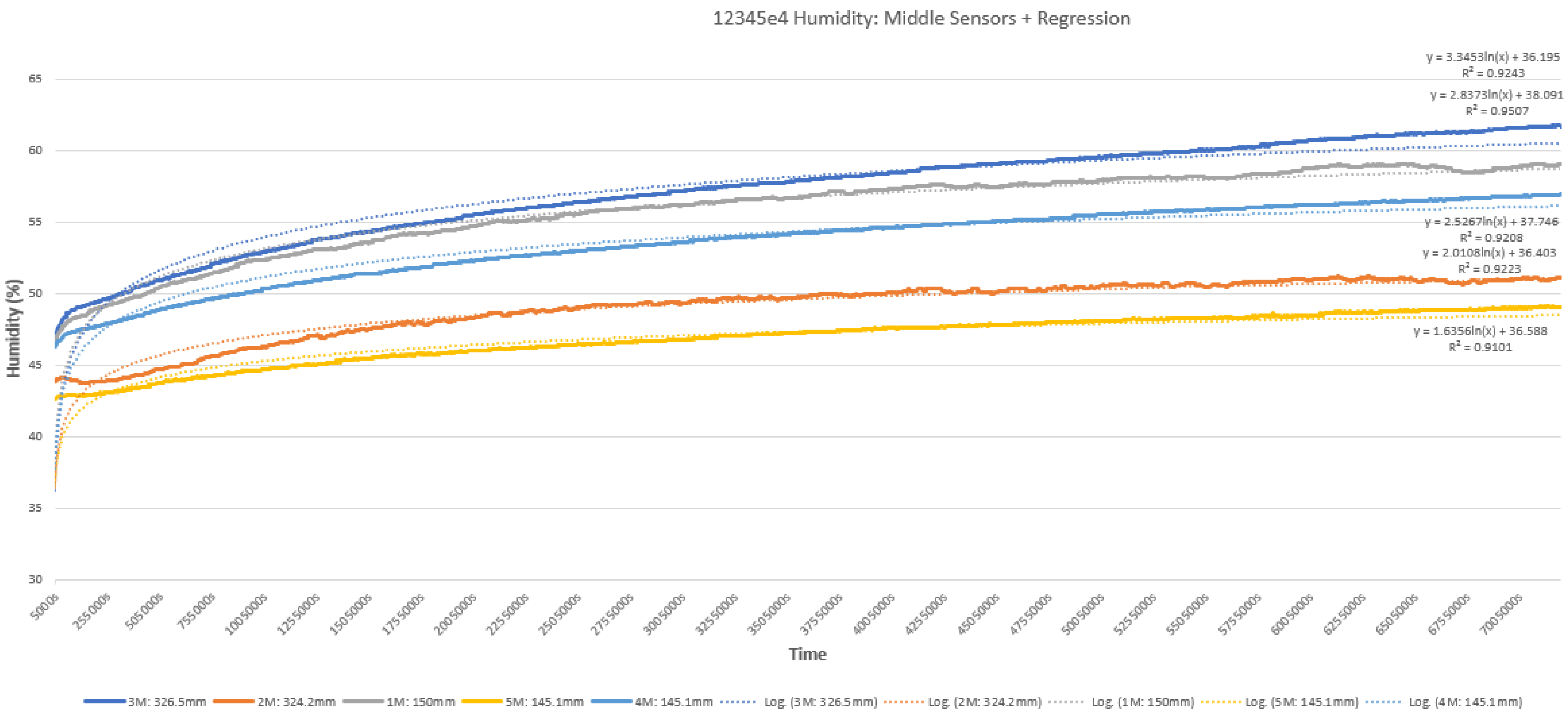


Figure 46: 12345e4 Middle Humidity Sensors + Regression

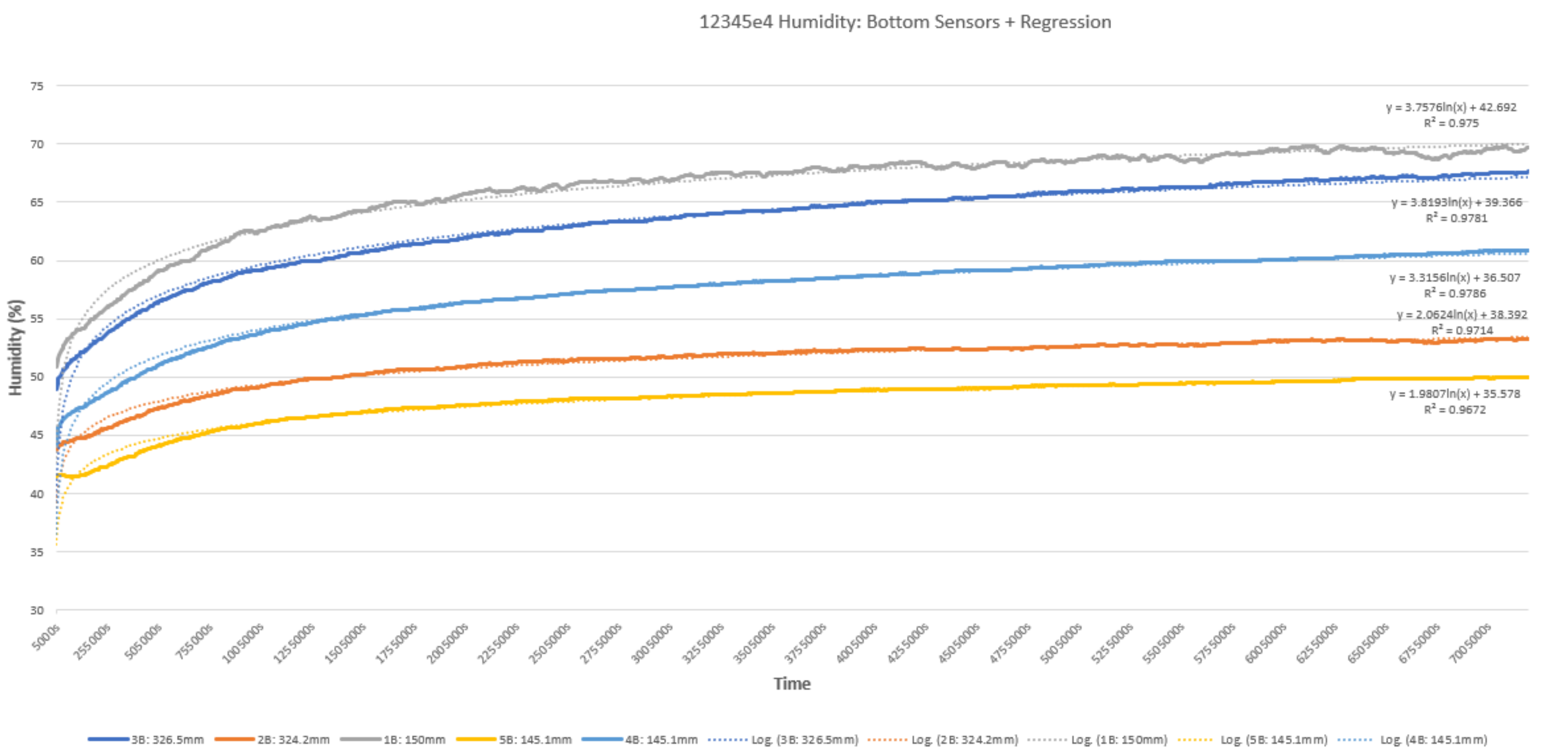


Figure 47: 12345e4 Low Humidity Sensors + Regression

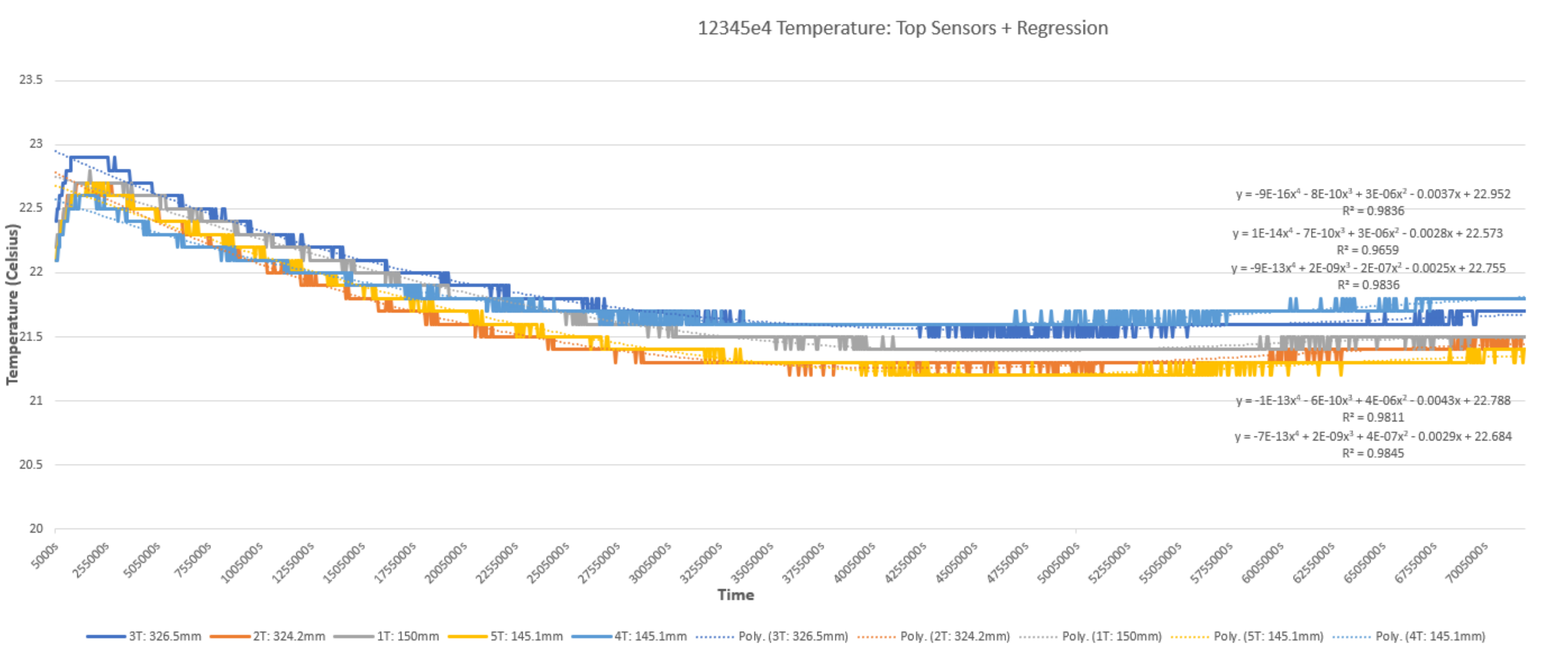


Figure 48: 12345e4 High Temperature Sensors + Regression

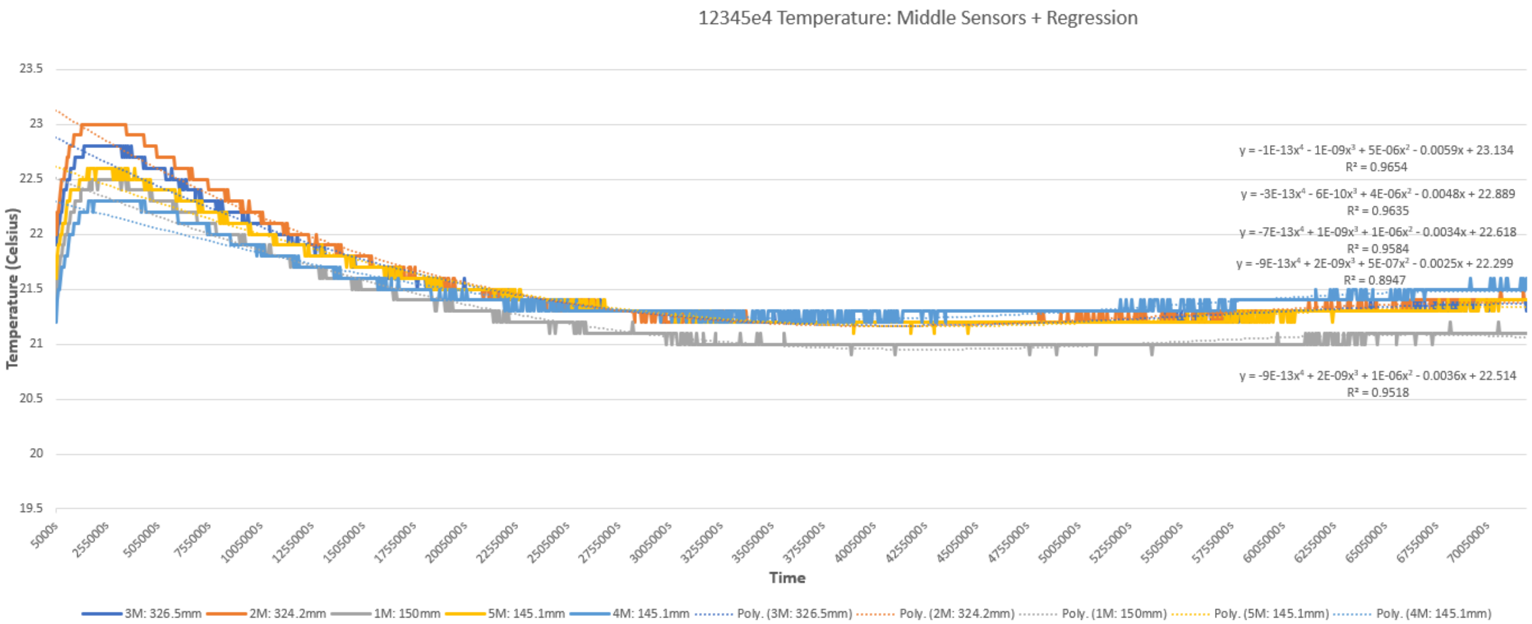


Figure 49: 12345e4 Middle Temperature Sensors + Regression

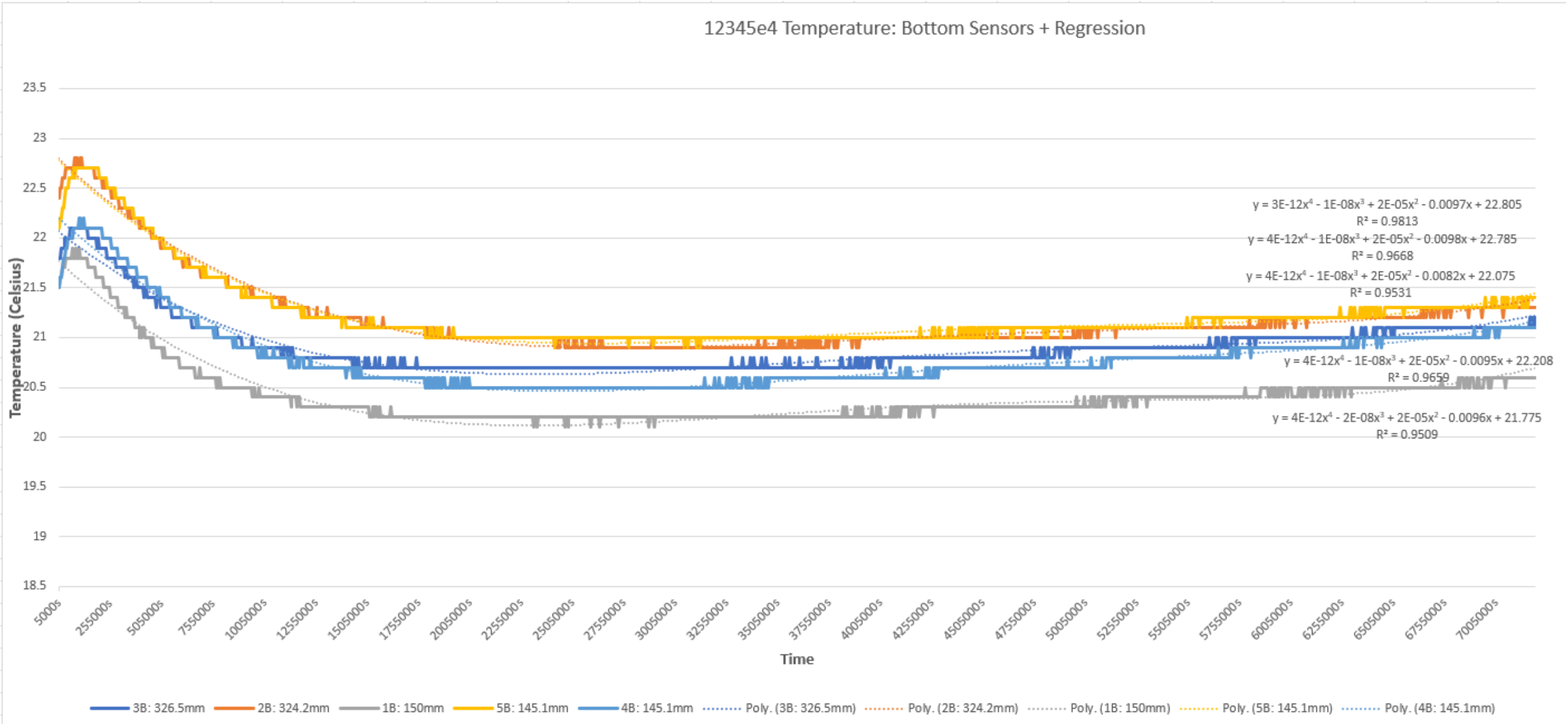


Figure 50: 12345e4 Low Temperature Sensors + Regression

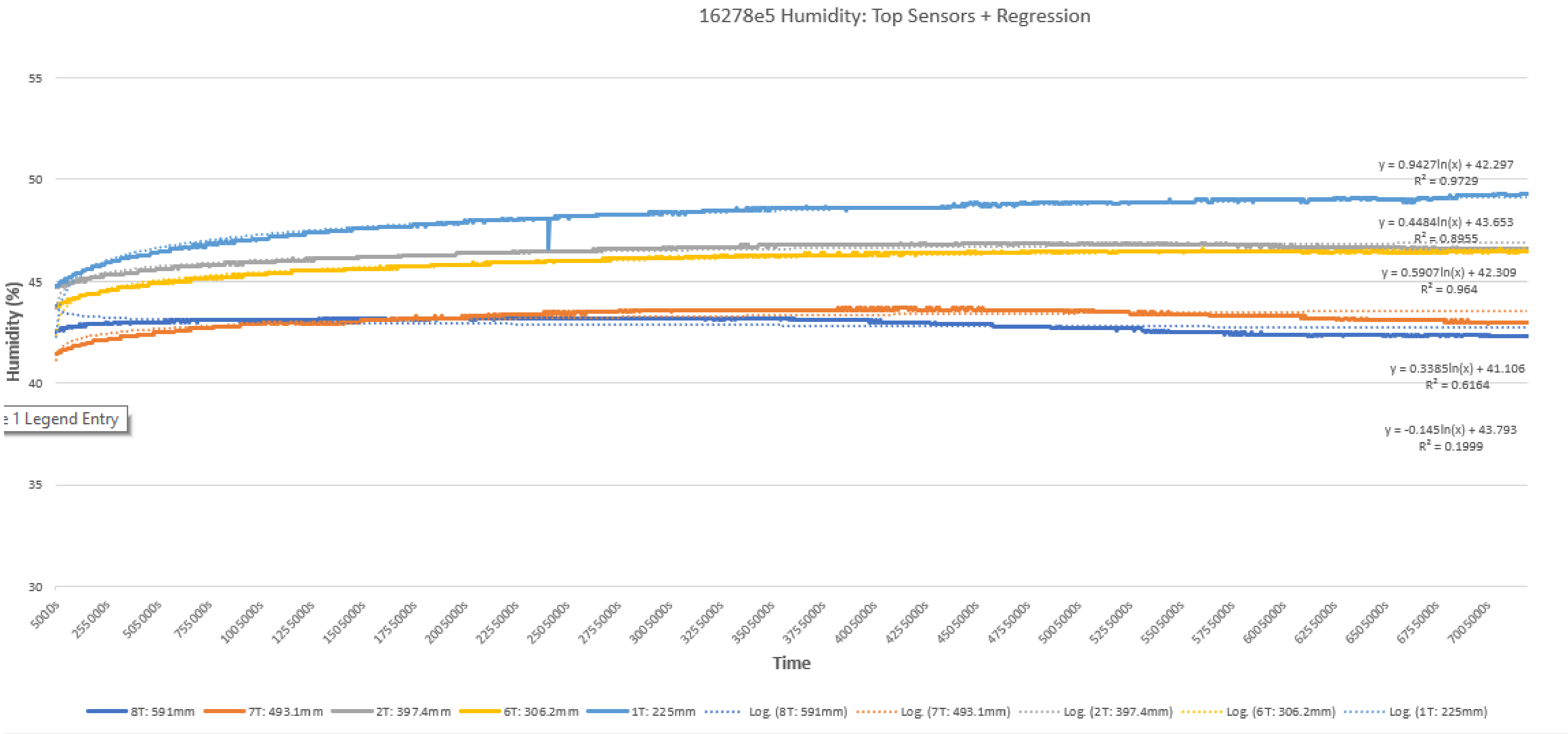


Figure 51: 16278e5 High Humidity Sensors + Regression

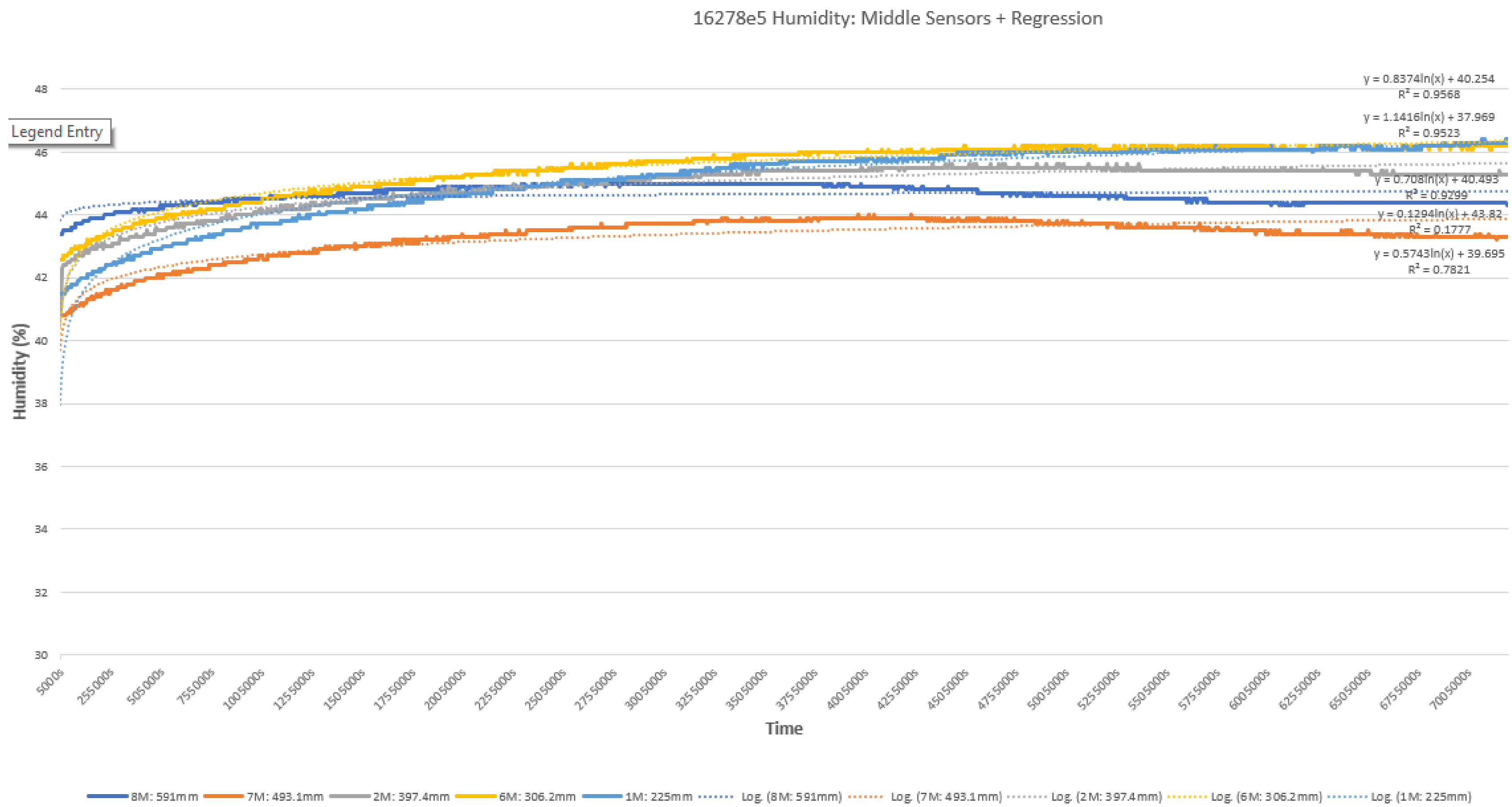


Figure 52: 16278e5 Middle Humidity Sensors + Regression

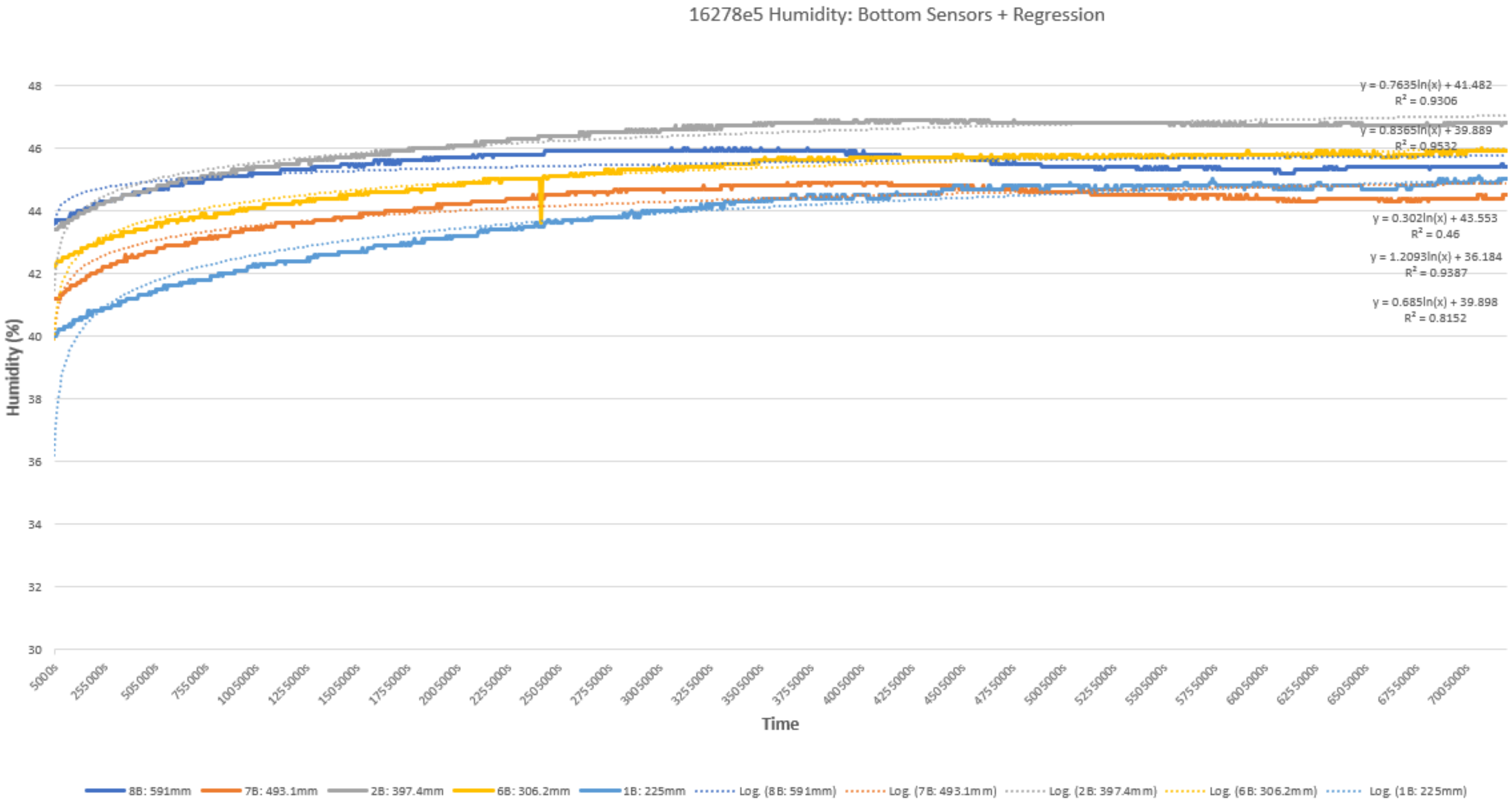


Figure 53: 16278e5 Low Humidity Sensors + Regression

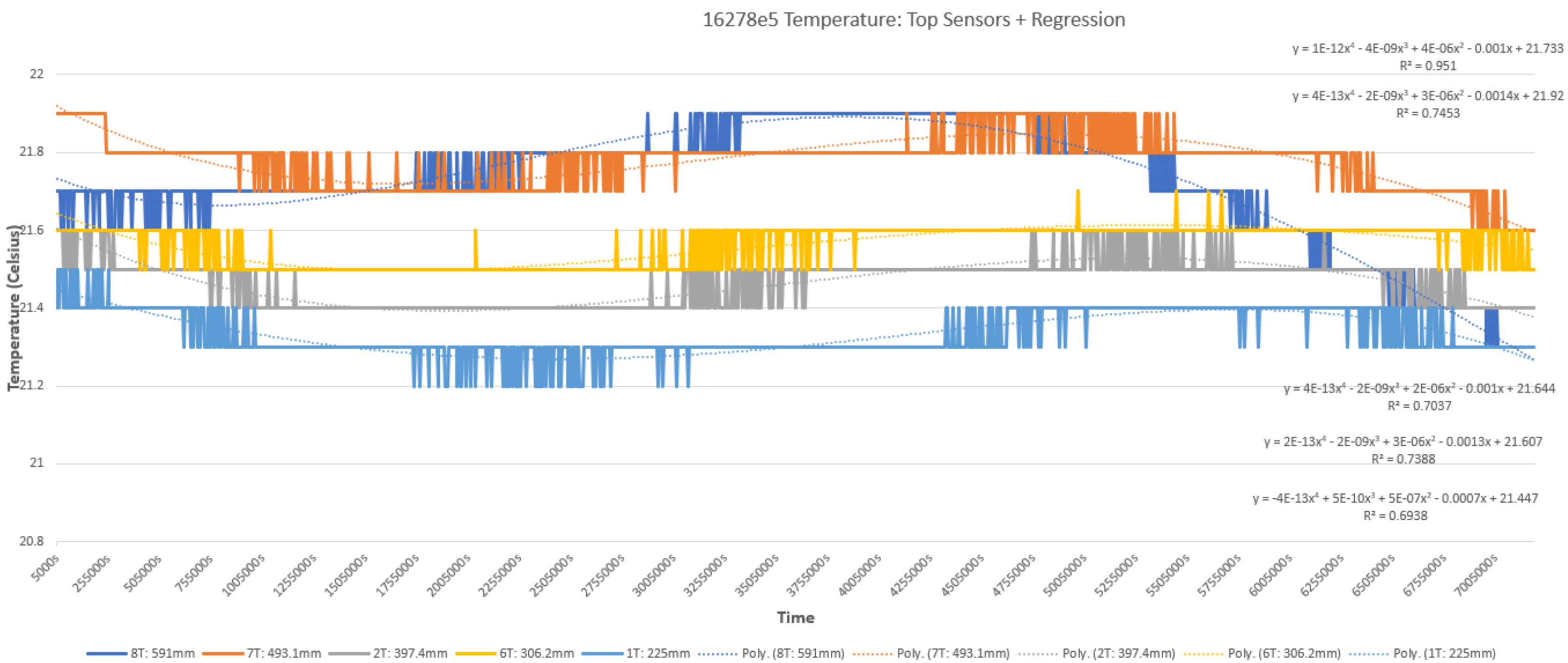


Figure 54: 16278e5 High Temperature Sensors + Regression

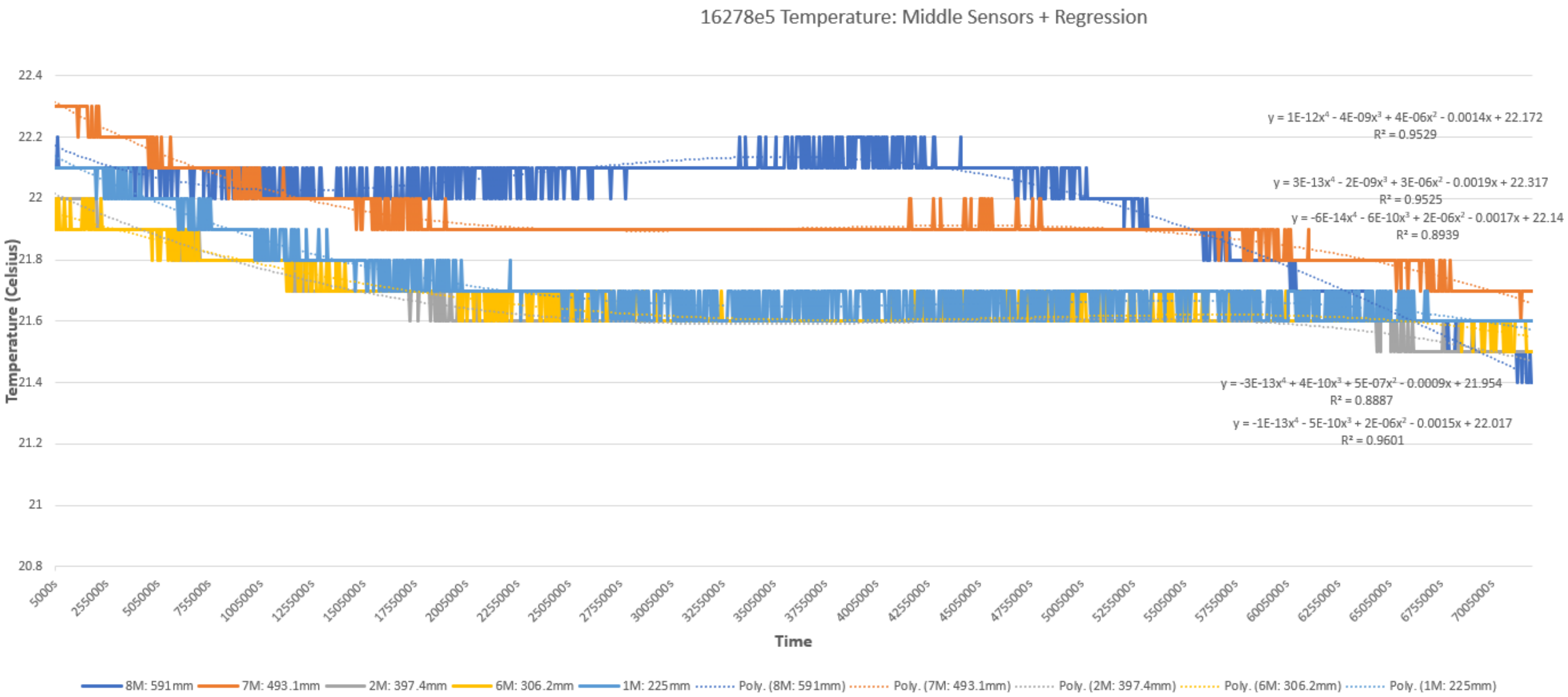


Figure 55: 16278e5 Middle Temperature Sensors + Regression

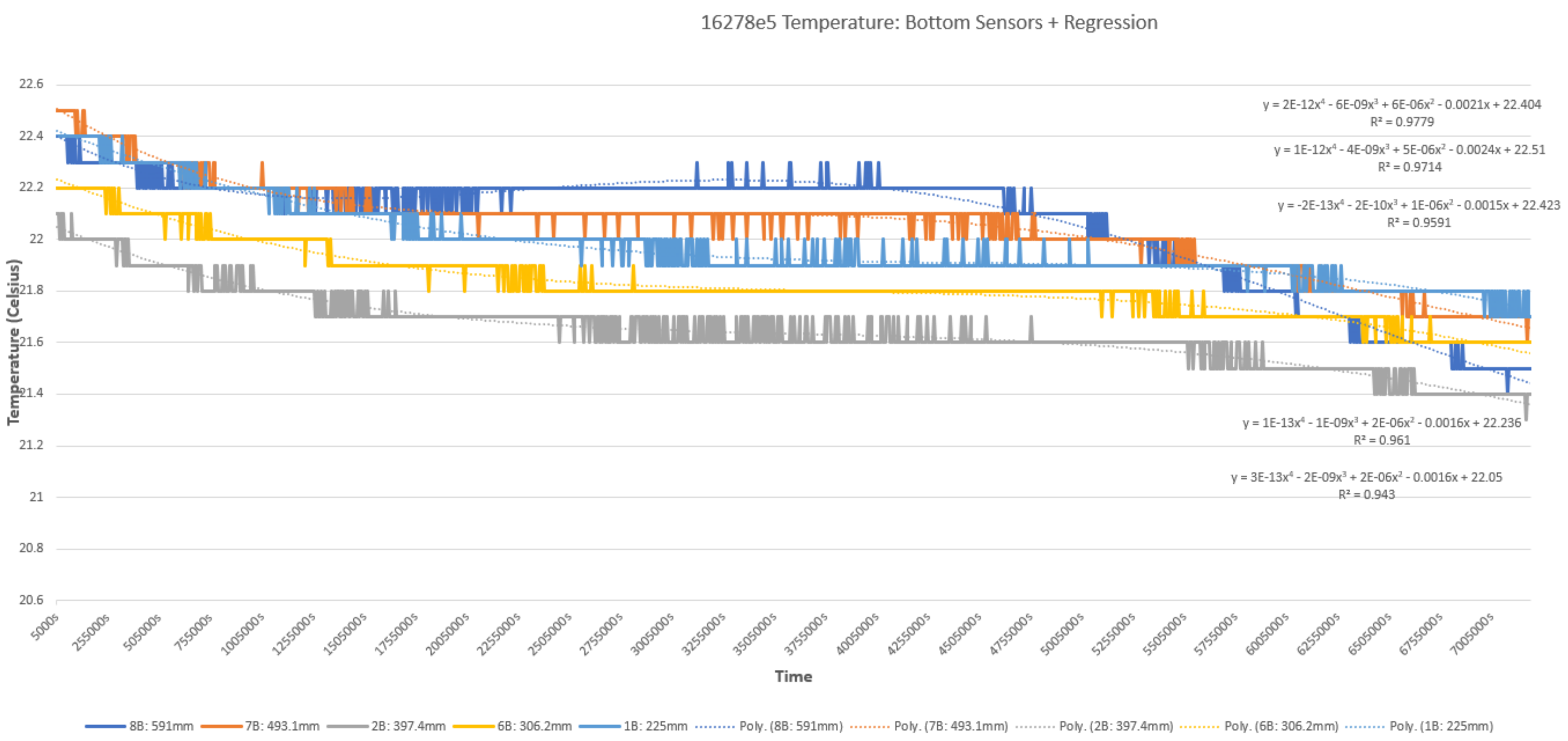


Figure 56: 16278e5 Low Temperature Sensors + Regression