Project description:

MeknoMonitor:

The MeknoMonitor is a monitoring device designed from the ground up to monitor and transmit data collected from sensors regarding the operation of the Paladin – Meknology’s patented technology to sustainably desalinate water. The MeknoMonitor monitors temperature and pressure from critical points in the system to give a readout to users both locally and across the globe. This project was implemented in 3 main sub projects. The monitoring console, the web app monitor, and the Real Time Database (RTDB) hosted on Google’s firebase cloud computing software.

The Console was the largest part of the system, “it was much more than we need a box, let’s build a box.”-Eric LeBlanc. In the development of the console, we made an effort to mimic the product design process we used for the paladin as a whole. This included Failure Modes Effect Analysis (FMEA), materials consideration processes, process flow diagrams, Rapid Iterative Testing Evolutions (RITE), and many other considerations to ensure that the console was able to perform to our standards and give us the results we needed.

Google’s cloud computing software, Firebase, was used to host the database due to its ease of use, overall cost, and it’s integration into our current ecosystem. We considered using other services such as AWS, IBM Cloud, and Azure Cloud, however, after running a cost benefit analysis, service analysis, and integration consideration, we decided to go with Firebase.

The web monitor was an html page that fetched the current readings cyclically using an html-json request. The backend was kept simple to improve access time, and the UI was coherent with the main website identity.

Console:

The console was modeled after a modified operator’s consolette. This shape & design was chosen after research into the environment of the final product. This console’s shape is ideal for multiple mounting positions, either on a wall or podium, and the angle of the lid allows for multiple points of view. The placement of the console with the paladin was difficult, as wire length, physical location of install, power & internet connection requirements were all concerns that restricted the placement of the console.

Determining what measurables needed to be monitored was a time consuming but rewarding process. Using the FMEA, Daniel, Eric, and the representative from a sensor distributor, we were able to look at every possible failure mode, and determine a way to monitor it, with just 3 sensors.

Selection of materials for the console involved water penetration tests, electrical conduction tests, cost analysis, source analysis, and ecological impact analysis. The results of these tests showed that Masonite, an engineered wood that consists of layers of wood fibers glued & pressed together. I worked with an intern to design a 3d model of the console. This model was then sent to the prototype lab to be “flattened” – a process that takes the 3d model and splits it into 2d panels that can be cut out with a CNC machine and pieced together.

Once the model was flattened, it was uploaded to the lab CNC machine, where it followed the cutout pattern. The console was then reassembled, and we had a physical prototype for the console ready for the insides to be implanted.

The electronics inside the console had to go through a rigorous selection process as well. This device will be operating “away from home” most of its life, so a robust and reliant system is the highest priority. With our stock of hobby sensors and micro-controllers, we needed to ensure that the system was repairable in the event of an emergency, as the failure rate with these sensors is much higher than standard industrial sensors. Because of this, we decided to keep the system on a standard breadboard and use standard breadboard connections instead of soldering components directly to our prototype board.

We decided to use an Arduino uno R3 as the brains of the console. This micro-processor is extremely versatile, and has safeguards built into the board level preventing over current, shorts from many points, and fused power to provide an extra layer of protection. In addition, the expandability of this board allows us to break it out and use the processor independently allowing future expandability. The massive network of hobbyists is growing daily, so any issues that arose, a reference for that problem was already created, as someone has used that part and had a similar issue.

We used the MPRLS Ported Pressure sensor. This sensor is a hobbyist sensor that has 6 pins, only needing 4 for standard operation. This sensor operates on the i2c bus, a modern communication protocol that uses one wire for a clock pulse, and another for signed data. Each module connected to the i2c bus has its own address and is able to communicate with the micro-processor. This sensor has many advantages; however, the one common downside is their lifespan. when measuring dry pressure there is no issue with durability. However, considering the material we are measuring, the lifespan of these sensors degrades rapidly. In fact, we saw the degradation of one sensor a week before our final demo day due to constant testing and needed to have a replacement within the week.

The temperature sensors we used were much more durable. We chose Resistance Temperature Detectors (RTD) Thermistors, as their simplicity allowed us to incorporate multiple with ease, and we did not need to worry about damaging these as much. The resistance from the sensor probe was returned to the micro-processor for interpretation, and display. The nature of this sensor means that we need to compensate for the length of the sensor wire, and since we were limited in length, this put a constraint on the placement of the console in regard to the paladin system. Because of the wire length, we decided to place the console just above the opening for the tube. This allowed easy viewing from the height of the system, as well as prevented accidental water spillage.

One key function of the MeknoMonitor was the ability to cut the power to the system as a whole. Due to time constraints, the pump controls were not integrated into the console, however, the power to the box was controlled by a toggle switch that would turn on and off the monitoring system.

Two buttons were added to the exterior of the MeknoMonitor, one to trigger the pressure reading on the display, and one to show the temperature reading. Each button was wired directly to the Arduino, and when pressed would swap the display mode. The console still receives both values within one JSON request and prints both to the serial monitor for diagnostic & efficiency purposes.

A 16x2 LCD Display was added to show the two readings. This size screen was the smallest where the information was still legible. This display initially had capabilities to operate on the same i2c bus as the pressure sensor, however after an issue with the i2c backpack, the display was wired manually.

A 9volt battery pack was included inside the console to operate as a battery backup if needed. During the final demo, the console operated on this battery pack for 5 consecutive hours with no issues.

Firebase:

The server side of this project was relatively painless. There are many resources for uploading data from the Arduino, as many hobbyists use firebase for their home improvement projects. Firebase’s Real Time Database is an application interface that stores data in real time, and synchronizes application data for IOS, android, Web, Linux, mac and more.

This platform’s cross compatibility, future expandability, and price point made it the clear winner over the other services under consideration, Amazon Web Service, IBM Cloud, and Microsoft Azure. The non-SQL nature of the RTDB means that there will be lower latency, a larger data headroom, and a somewhat more flexible data model.

The setup with this application is relatively easy. An API key, and database URL are the only things needed to set up a free database, which can be allocated through firebase. The console that will be transmitting data has to package the data up for transmission as a JSON packet. The formatting for this can be found in the firebase configuration guide, and there are countless YouTube videos with instructions, tips, and tricks.

Operating on a 5 second cycle.

Upload procedure:

First, the console checks if the database is online. (.5 seconds)

Upon confirmation, console uploads data (pressure & temperature) (1 second)

RTDB returns a verification number upon successful upload. This verification number is the sum of both readings divided by 2. The console performs the same operation on the number internally, if they match, no action is taken. If the numbers do not match, the next packet will send a flag to indicate the previous reading was not correct. (.5 seconds)

.5 second break

Website:

The website was built to monitor the variables passed to the RTDB in real time. This was a single page website made in html with two bars, one representing temperature, and one for pressure. These bars give a visual indication of the pressure and temperature, as well as the numerical readout below the bar. This webpage was designed to match the design of the Meknology website. Colors were chosen to match the ones used in the site, the same font was used, and the standard slogan was placed on top.

Download procedure:

Website confirms database is online. (.5 seconds)

Upon confirmation, website reads data from RTDB (pressure & temperature) & checks for flag to display. (1 second)

Website writes values from json packet to website (.5 seconds)

1 second break

repeat