**Scale Shelf Development**

The scale shelf was constructed from the following items:

**Scale:**

100mm x 45mm x 10mm pieces of softwood x 2

Nut and bolt x 2

55mm x 12mm x 12mm 5 kg Load Cell Bar x 1

**Shelf storage bin:**

76mm x 101mm x 167mm plastic stackable storage bin x 1

**Load Cell Amplifier:**

Sparkfun HX711 Load Cell Amp Breakout Board x 1

**Microcontroller and IDE:**

Arduino Uno R3 (Initially)

Arduino Uno Wifi R2 (For most of development and final product)

Arduino IDE version 1.8.12

**Miscellaneous:**

70mm x 50mm Perf board x 1

Jumper wires x 4

100mm x 20mm Velcro strips x 4

5 pin Header pins x 2

**Scale Construction:**

The pin holes on both sides of the HX711 were soldered to the perf board with the header pins. Each of the red, black, white and green wires coming from the load cell was then soldered to the perf board and connected to the RED, BLK, WHT, GRN pins of the HX711 by means of solder bridges. The same technique was used to connect jumper wires to the VDD, VCC, DAT, CLK, GND pins on the other side of the HX711. The information on how to connect the load cell and the Arduino to HX711 was gained from the online Sparkfun HX711 hookup guide2.

Two pieces of wood were cut to create a wooden base and scale platform. A hole was drilled in the front of the platform piece and the back of the base piece in order to attach the load cell to both in such a way that any weight placed on the platform would cause the load cell to pivot downwards causing a strain that can be detected by the strain guages in the load cell, amplified by the HX711 and then sent to the Arduino. The load cell was attached to the platform and base with nuts and bolts. The nuts also served as spacers between the platform/base and the load cell. Finally the plastic storage bin was attached to the scale platform by means of velcro strips. The strips keep the bin stable on the scale but allow for the bin to be removed if required (e.g. for cleaning). The scale design was based on the online Instructables tutorial on how to build Arduino weighing scales1.

**Connection to Arduino:**

Once the basic construction of the scale was completed it was ready to connect to an Arduino microcontroller for calibration. Initially an Arduino Uno R3 was used, which does not have built in wifi or ethernet capability. This was adequate for the initial set up and calibration of the scale. The Arduino was connected to an Acer Laptop via a USB cable. The laptop operating system was Ubuntu Linux 18.04 and it also had the Arduino IDE version 1.8.12. installed on it. The demonstration Arduino sketch from the instructables guide1 HX711.ino was downloaded and stored in the IDE and the required Arduino library HX711.h was added to the IDE3. The demonstration sketch was uploaded onto the Arduino itself which was then ready to be connected to the scale.

The scale was connected to the Arduino using the jumper wires previously connected to the HX711 in the following way:

|  |  |  |
| --- | --- | --- |
| **HX711 Pin** | **Jumper Wire Colour** | **Arduino pin** |
| DAT | White | A0 |
| CLK | Orange | A1 |
| GRD | Green | GRD |
| VCC | Black | 5V |
| VDD | Green | 3.3V |

Once the scale and Arduino and laptop were all connected together with the HX711.ino sketch running, it was possible to view scale weight values processed and output in the Arduino IDE Serial Monitor window. An intial glance at the output confirmed that the scale was providing credible looking data, but to be sure the scale was accurate it next needed to be calibrated.

**Scale Calibration:**

(This method comes from Step 8 of the Instructables guide1.)

The HX711.ino sketch includes the line:

scale.set\_scale(2020.f);

which sets the calibration value. The preset value of 2020 in the sketch is designed for obtaining readings in grams from a 1kg loadcell. In order to calculate the calibration value required to obtain readings in grams from a 5kg load cell the following steps were followed:

1. Start with an arbitary calibration value (I retained 2020)
2. Obtain an item of known weight that is at least a tenth of the scale’s maximum weight (I used a 500g package of dried macaroni) which was weighed on a post office scale to obtain a reliable weight value of 507g
3. Place item on the scale and obtain a weight reading
4. Multiply current scale factor by the weight reading, then divide by the known weight obtained in Step 2. The value obtained is the new calibration value.
5. Repeat the steps using the new calibration value in Step 1 until the weight readings obtained at Step 3 are accurate.

After following this process, a calibration value of 458 was calculated. Other items such as 9V batteries and tins of baked beans were then weighed on the scale and the values obtained checked with the item weights obtained from kitchen scales to ensure the scale was measuring consistently and reliably.

**Sending weight data to the web application:**

The next task was to get the weight readings calculated by the Arduino into the web application. This would entail sending the data either by Ethernet or Wifi to the Computing Department network where our webserver and database would be running. Although an Ethernet connection would have been a more reliable option, it was problematic for two reasons. The first was that an Ethernet module would have had to be obtained and soldered to the Arduino R3. The second was that permission would be required to have the Arduino ‘whitelisted’ on the college network and configured for access to the Computing Department virtual servers. As development time was short, it was decided to pursue the wifi option instead.

**Network connectivity:**

The Arduino Uno R3 was replaced with an Arduino Uno Wifi R2 microcontroller which has a built in Wifi capability, and the required WiFiNINA.h library was added to the Arduino IDE. An initial test of the Arduino wifi connection was accomplished by writing an Arduino sketch which would attempt to connect to a home wifi network and then output information such as the IP address into the Serial Monitor of the Arduino IDE to signify success or an error message to signify faliure. The information regarding how to write this sketch was gathered from an YouTube video tutorial5. The Arduino was able to successfully connect to the home wifi network.

Next an attempt was made to connect the Arduino to the Goldsmiths eduroam network. Unfortunately this was not possible as the Arduino connects to wifi using just an SSID and a password whereas eduroam is a WPA2 Enterprise network which requires a username to be input as well as an SSID and password. Attempts were also made to connect to the Goldsmiths guest network (which had the same issue as eduroam). Being unable to connect to the network while on campus at this point was detrimental to the scale development process as a lot of time and effort was spent trying to find a solution to the issue and it meant that a lot of work on the scale carried out on campus could only be evaluated when the developer was at home, which slowed the process down considerably. It also led to integration of the scale with the web application taking place at a much later stage of development than was desirable. Towards the end of the the development process a workaround was discovered whereby the Arduino and Laptop could both be connected to the Department of Computing, Hatch Labs G11 wifi network which only requires an SSID and a password. This was helpful as it enabled more development to take place on campus and would have meant that integration work could have taken place there. Unfortunately the Covid-19 pandemic followed soon after and all further development had to take place at the developer’s home.

**Attempt to use wifi and POST method to send data to our webserver:**

There were two choices regarding how and where to send the weight data from the Arduino. The first was to send the data using HTTP and POST to the Node.js/Express webserver to be processed in the middleware layer of the web application. The second was to send the data directly to the MySQL database using tools from an Arduino MySQL Connector library, where it could be retrieved and processed by the web application.

A test using adapted Arduino sketch code from a YouTube video tutorial on temperature information from an Arduino to a webpage5 worked successfully for outputting random numbers and then scale weight values to the localhost/browser on the developer’s laptop, using the Arduino as it’s own server. This method did not however use HTTP/POST, which would be required for sending the data to a Node.js/Express webserver. An attempt was made to adapt code from a second YouTube video tutorial which demonstrated how to send data from an Arduino Wifi by POST to a webserver4. The developer could not successfully get the group Node.js/Express webserver to accept the data through HTTP/POST and there were also issues with permissions to access the virtual server ports. As the tutorial did not specify the type of webserver used (it was probably Apache) and the server side scripting language used was PHP, the developer did further research on how to resolve these issues for Node.js/Express/Javascript. As development time was running out however and progress was slow, it was decided to explore the option of sending the data directly into the MySQL database instead.

**Use Wifi and MySQL connector library to send weight data direct to a database**

An excellent tool for getting data directly from an Arduino to a MySQL server is the MySQL Connector/Arduino library created by Dr Charles Bell6. In addition to the scale, Arduino and laptop configuration previously described, the following components were required:

* MySQL Connector/Arduino Library installed in Arduino IDE
* A MySQL 8.0 server (with legacy authentication) was set up on the developer’s laptop.
* A database called test\_arduino was created
* A user account was created with native authentication and granted all privileges to access the test\_arduino database
* Access to port 3306 was enabled on the laptop
* The MySQL server and the Arduino must be connected to the same wifi network segment.

Once everything was set up, a basic test of the user account direct access to the test\_arduino database on the MySQL server was conducted both on the developer’s laptop and from a different laptop on the same home network. This was followed by a test SQL INSERT statement directly into a table created in test\_arduino to ensure that the weight type data (float type) was inserted and represented correctly. Both of these tests were successful.

The next challenge was to connect the Arduino to the MySQL server on the developer’s laptop. In order to do this an adapted version of the example sketch Listing 2: Sample Connection Test – Wifi from the MySQL Connector/Arduino User Guide was used6. Initially there was an issue whereby, after the Arduino had connected to the wifi, the sketch would hang at the MySQL server connection stage. Some research online revealed a forum discussion between a user with the same issue and a response from Dr Bell with some very useful suggestions involving adding some extra libraries to the sketch and commenting out some code in one of the MySQL connector cpp files7. This resolved the issue and it was then possible to connect to the MySQL server with laptop and Arduino both on a home network.

Now that it was possible to connect to the database server from the Arduino, the ability to send an INSERT statement to a table in the test\_arduino database was tested. The ability to send a static query string of either integer or float data was successfully tested using an adaptation of Listing 3: Simple Data Insert Sketch from the MySQL Connector/Arduino User Guide6. Sending readings such as weight data from a sensor however requires that a new query string is built each time data is sent. This issue is addressed in Listing 4: Complex Insert Sketch from the MySQL Connector/Arduino User Guide6. This sketch was also successfully modified to insert regular weight readings from the scale into a table in the test\_arduino database.

**Putting it all together:**

A first draft of an Arduino sketch that combined setting up the scale, processing weight readings connecting to wifi and the database server and inserting readings into a database table was created. The sketch combines and adapts parts of all the example sketches previously mentioned1,3,4,5,6. A database dump file of the project database was used to create a replica of the application database on the developer’s laptop. The draft sketch was modified to insert data into one of the shelf tables (id1weights) in the replica application database and run while adding or removing items of known weight to and from the scale. This was a successful test of the integration of the database and the Arduino scale. A quick and dirty test of the integration of the components above with the latest version of the web application(running on the developer’s laptop) was carried out by checking if the weight values being sent directly to the database were being correctly detected, processed and displayed by the web application. This was successful in that changes to weights caused by adding and removing items to/from the scale bin were correctly represented by the web application shelf details for shelf 1 when the web page was refreshed in the browser.

The draft sketch was reviewed at this stage and the following improvements were made:

* The sequence of activities was placed in a more logical order
* Redundant or commented out code was removed
* Each section was commented
* Code was written into the loop section of the sketch to check and resolve the status of the MySQL server connection before attempting to send data (as the sketch would hang if data was sent when the connection had dropped).
* A time delay was added to control the rate at which weight data was sent to the database

**End of development phase:**

As the development phase of the project was now drawing to a close and the scale was successfully sending data to the application database via the Arduino it was decided that development on the Arduino scale would halt here and attention would now be focused on formal testing of the integration of the Arduino scale, database and web application. For details of the testing phase, please view the Quality Assurance section of this report.

**Next steps:**

* **Connectivity:**
  + Use an Ethernet enabled Arduino and have it whitelisted for the Goldsmiths network and also with permission to access the virtual servers of the Department of Computing. This would hopefully be more stable,reliable and secure than a wifi connection.
  + Try to resolve issues encountered with accessing the database server on our virtual server space. Retry suggestions provided by Eamonn Martin regarding changing the port setting in the MySQL server config file to 8000, restarting MySQL server and changing the MySQL port to 58524 in the sketch I tried this with the Arduino on the G11 network and it didn’t work. The MySQL Connector Arduino user guide states that the Arduino and the MySQL server must be on the same network segment? It could however be an issue with how to correctly express the server address and port of the group virtual server in the Arduino sketch.
  + Investigate the possibility of using a Raspberry Pi instead of an Arduino. The web server and database server could then possibly run on the Pi as well.
* **Sending data by HTTP/POST to web server:**
  + Do further research on how to send data by HTTP/POST to the web server. I found a solution online which creates a Node.js/Express HTTP server and uses web sockets to communicate with JavaScript embedded in an HTML file. I think this solution seems unecessarily complicated though and perhaps just more research is required on how to send and handle a POST request outside of a browser using Node.js/Express/JavaScript
* **Scale Shelves:**
  + Try and develop a scale that can detect very light items individually. A much more expensive load cell would definitely be required for this.
  + Build enough scale shelves for all six shelf slots in the web application.
  + Investigate if all six shelves can share one HX711 amplifier and one Arduino.
* **Sketch:**
  + Research and write code that will automatically restart the Arduino if the wifi connection drops.
  + Research and write code that will allow the web application to send instructions to the scale e.g. to reset a shelf or swap shelves.

**References:**

1. Instructables Tutorial: Arduino Weighing Scales: <https://www.instructables.com/id/How-to-Build-Arduino-Weighing-Scales/>

2. Sparkfun Tutorial: HX711 Hookup Guide: <https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide>

3. HX711 Arduino Library: <https://www.arduinolibraries.info/libraries/hx711-arduino-library>

4. Eli the Computer Guy You Tube Video Tutorial: Write POST Data to Server with Arduino Uno with Wifi: <https://www.youtube.com/watch?v=32VcKyI0dio&list=PLJcaPjxegjBUsCc8PDvalF9j9dvc1RpUh&index=26>

5. Eli the Computer Guy You Tube Video Tutorial: Arduino Wifi Temperature Web Page Alert: <https://www.youtube.com/watch?v=4IhSCSPQMA0&list=PLJcaPjxegjBUsCc8PDvalF9j9dvc1RpUh&index=28>

6. MySQL Connector/Arduino Library: <https://github.com/ChuckBell/MySQL_Connector_Arduino>

7. Issue in conn.connect(server\_addr, 3306, user, password) Arduino uno wifi rev.2: <https://github.com/ChuckBell/MySQL_Connector_Arduino/issues/80>