THESIS

# PROOF OF CONCEPT OF AN AUTOMATED COCORAHS RAIN GAUGE

Submitted by

Jieqi Lin

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2016

Master's Committee:

Advisor: Gearold R. Johnson

Wade O. Troxell
Thomas J Siller

# ABSTRACT

## PROOF OF CONCEPT OF AN AUTOMATED CoCoRaHS RAIN GAUGE

CoCoRaHS is the Community Collaborative Rain, Hail and Snow Network, collecting daily precipitation reports from volunteers' reading of a low-cost plastic 4-inch diameter rain gauge. Nolan Doesken, the Colorado State Climatologist following the disastrous Fort Collins flood of July 1997, created the network. Today there are more than 20,000 rain gauges in service. There are three limitations in the present CoCoRaHS rain gauge. The first limitation is limited temporal granularity, i.e., data measurements are only available once each 24-hours, in the best cases. The second limitation is not being able to capture data during an extreme event as it is occurring. The third limitation is because this is a volunteer activity other activities may interfere with the ability to collect daily data. The question is can the CoCoRaHS rain gauge be modified to eliminate these three restrictions while keeping the design as similar to the current system as possible.

The design is to turn the present rain gauge into a weight-based automated rain gauge. An in-lab tested prototype was built requiring only that the rain gauge bracket needed modification. The result is a modified bracket holding a load-cell, microcontroller and Wi-Fi communications subsystems. A number of experiments were designed and conducted to prove the feasibility of the rain gauge. The conclusion is that a low-cost, high-accuracy weight-based automated CoCoRaHS rain gauge is achievable.

# ACKNOWLEDGEMENTS

The research for this thesis would not have been possible without the support and guidance of numerous people, to whom I am very grateful.

I would like to express my deepest gratitude to my advisor and mentor, Prof. Gerry Johnson, who has guided me through my M.S. study and research with enthusiasm, patience, and wisdom. I am very grateful to him for affording me many incredible opportunities, guidance, constant encouragement, constructive criticism, and enormous patience throughout all these years. I am extremely fortunate to have had Prof. Gerry as an advisor, colleague, and friend and can never thank him enough.

I would like to extend my thanks to my M.S. committee, Prof. Wade Troxell and Prof. Thomas Siller, for their advice, insight, and efficient guidance. They have been of great help in assisting with and supporting my research and academic study. Also I offer particular acknowledgment and gratitude to MetStat, Inc. and CoCoRaHS.

I thank Douglas Hopper, Guirong Liu, Lang Yang, Derrick Zhou, and Ce Guo, who provide their strong technical support, many interesting discussions and practical suggestions as I learned many new things. Without that help, this work might not have been completed.

Last but not least, I am grateful to my parents, Nengwen Lin and Huanyi Zhu, for their love and support. My family has been encouraging, supportive and shown belief in me and my work. I could not have accomplished this without them.

TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction to CoCoRaHS

### 1.1.1 General

CoCoRaHS is the Community Collaborative Rain, Hail and Snow Network, which is “a unique, non-profit, community-based network of volunteers of all ages and backgrounds working together to measure and map precipitation (rain, hail and snow)” [1]. By using a low-cost measurement rain gauge device, stressing training and education, and utilizing an interactive Web site (www.cocorahs.org), CoCoRaHS was developed to provide quality data for natural resource education and research applications. Volunteers of CoCoRaHS cover all fifty states in the United States and are expanding around the world.

### 1.1.2 History

The idea of CoCoRaHS originated following a devastating flash flood that hit Fort Collins, Colorado in July 1997. According to Henry W. Reges, over 14 inches (355 mm) of rain fell in several hours flooded some areas of the city while other portions had only modest rainfall [2]. The rainfall from July 27, 1997 to July 28, 1997 (see Figure-1) of over 14 inches of rain fell in just 7 hours on the west side of Fort Collins, while only 2 inches fell just 5 miles to the east. The existing sources of precipitation data in 1997, such as meteorological radar, satellite data, official weather stations and spotter reports, failed to accurately capture the intensity and extent of the highly localized storm. The flood alarm was not sent in time and the heavy flood ravaged Fort Collins.

According to Bill Murray, this was a 500-year flood for Fort Collins. Damages on Colorado State University campus were surprisingly high during this flood [3]. The Morgan Library was inundated and more than 425,000 water logged books had to be shipped to a giant freezer to prevent mold damage; about half of the books were salvaged. Research documents, journals, and historical records were lost. Over 40 buildings were damaged and some graduate students were displaced from student housing [4]. Five people died despite valiant rescue efforts during the flood. Damage totaled over $200 million.

Through a post-storm survey by the Colorado Climate Center, the storm was evaluated in quantity, intensity and spatial pattern of precipitation. The large rainfall amount in the western part of the city, and at a higher elevation, let the flood waters flow into the eastern part of the city. People realized the important role individuals could play in measuring, mapping and reporting precipitation amounts. However, there were few weather stations and rain gauges were located far apart. This inspired Nolan Doesken, who is the founder of CoCoRaHS and the state climatologist for Colorado at the Colorado Climate Center at Colorado State University, to build up a grassroots volunteer network of backyard weather observers to expand the present weather station network.


Figure-1 Rainfall (inches) for Fort Collins at 4 pm MDT July 27, 1997 through 11 pm July 28, 1997 (Distance between A and B = 5 miles, A=14.5 inches, B=2.0 inches) [5]

1.1.3 Main goals

In 1998, CoCoRaHS was begun. As stated on the CoCoRahs website, there are four main goals that CoCoRaHS tries to achieve. The first goal is to provide observers, decision makers and other end-users with accurate high-quality precipitation data on a timely basis. The second goal is to increase the density of precipitation data available throughout the country by encouraging volunteer weather observations, as well as by collaborating with existing precipitation networks. The third goal is to increase community awareness about our weather by inspiring and encouraging citizens to participate in meteorological science and to have fun doing so. And last but not least, is to provide enrichment activities in water and weather resources for teachers, educators and the community at large; thus, building a collective awareness of our climate and developing citizen's skills in scientific data collection.

### 1.1.4 Volunteer network

Volunteers are a vital part of CoCoRaHS. The perfect CoCoRaHS volunteer candidates are those who are interested in weather, scientists and others who want to help in the community. The duty of the CoCoRaHS volunteer is to spend a few minutes doing a “hands-on” precipitation measurement and reporting data to CoCoRaHS each day. Volunteers will be amazed at what they learn as they become more aware of the weather that impacts us all [6].

Through 4-inch diameter high capacity rain gauges, CoCoRaHS volunteers help provide critical precipitation observations by submitting standard daily observation of total precipitation to the CoCoRaHS website or phoning into the CoCoRaHS office [7]. If volunteers need to be on a trip for several days, they can enter their data as a multi-day report and leave comments if they think they know on which day precipitation fell. Besides, volunteers are suggested to keep a written record of their precipitation data, which is a backup in case CoCoRaHS has computer or phone problems. If a CoCoRaHS student intern spots a suspicious or erroneous report for one station, they may contact that volunteer to double check the data. CoCoRaHS recommends the member reading from 430 to 930 and prefer a 7:00 am reading. Reports from other times will not appear on CoCoRaHS maps.

Even though the 500-year 1997 flood was the inspiration for CoCoRaHS, the utility of CoCoRaHS data is not limited to extreme events. The observations are also used by the National Weather Service on a daily basis, as well as by Hydrologists, Emergency Managers and Water Departments. The data is important to city utilities, such as water supply, water conservation and storm water. The beaneficiaries could be as big as USDA-Crop production,

farm service agencies insurance adjusters, mosquito control, outdoor and recreation, or individuals such as engineers, ranchers and farmers. Members of CoCoRaHS can also benefit from the hands-on science of precipitation measurement and use the data as an educational tool whereas professionals do not benefit from hands-on measurements but must rely on data provided by weather services [8].

CoCoRaHS is also used as a major component when setting the Drought Intensity levels in the U.S. Drought Monitor. So even when the volunteer reports “zero” for days on end in the Spring, it does make a difference [8]. CoCoRaHS data is also used in bias-correcting radar-based rainfall estimates. This helps improve rainfall analyses that are used to create River Flood and Flash Flood guidance products used by meteorologists [8]. As an example, nearly 80% of the snowfall reports received by the National Weather Service for a January 2013 storm came from CoCoRaHS observers [8].

CoCoRaHS network now covers all U.S. and parts of Canada. A very ambitious goal of the network is one observer in every square mile in urban areas and one in every 36 square miles in rural areas. For the U.S. this would result in a network consisting of approximately 238,000 rain gauges. Therefore, the current network represents a little less than 10% of the goal.

### 1.1.5 Equipment – rain gauge

According to the CoCoRaHS website, the official CoCoRaHS rain gauge is a low-cost plastic 4-inch diameter rain gauge. It is composed of four parts (see Figure-2): a funnel, a measuring tube, and a 4-inch diameter overflow tube, and a mounting bracket. “The funnel

directs the precipitation into the measuring tube and magnifies it by a factor of 10. This allows observers to report rainfall to the nearest 0.01" (one hundredth of an inch). The measuring tube, when full, will hold "one inch" of rainfall. When it rains more than one inch, the excess water collects in the overflow tube. ” The cost of this rain gauge is around $30 and volunteers can buy parts separately.


Figure-2 CoCoRaHS gauge


Figure-3 Ten-year Comparison

Even though some volunteers have other kinds of automated rain gauges in their back yards, they are requested to report the data coming from the official CoCoRaHS gauge in order to accurately compare CoCoRaHS reports. Through a ten-year rain gauge comparison experiments (see Figure-3) [9], the CoCoRaHS gauge has shown itself consistent and reliable. At CSU Campus weather station, there are five rain gauges continuously used to measure precipitation. Four out of five rain gauges are the typical professional rain gauges. They are NWS Standard Rain Gauge (SRG), 8’’ diameter Dual-traverse Universal Weighing-Bucket Gauge, 8’’ diameter Fischer-Porter weighing gauge with 15-minute paper tape punch block, and 8’’ diameter tipping bucket rain gauge. In the experiment, the data was compared between SRG (8’’) and CoCoRaHS rain gauge. The data from the other three professional

rain gauges were used to calculate the average precipitation, which was then used as a comparison with SRG and CoCoRaHS. The results indicated that the SRG and the CoCoRaHS rain gauges produced results within 3% of each other over the ten-year experiment.

### 1.1.6 Restrictions

After eighteen years, CoCoRaHS is mature. Although the expansion of CoCoRaHS network is impressive, there are three restrictions limiting CoCoRaHS data.

The first restriction is limited temporal granularity. CoCoRaHS only gets one data point from the station per day, at best, to make a precipitation map if the volunteer reads the gauge and reports between 4:30 am - 9:30 am. It is impossible to catch a rain event immediately with 24-hour reporting. The present CoCoRaHS precipitation is more static, which limits the use of the map. A dynamic precipitation map will be much more helpful in extreme events.

The second restriction is reliability of precipitation data. People who collect and report data are volunteers from all ages and backgrounds. They have different jobs and different schedules. It is impossible to ensure every CoCoRaHS volunteer reports daily. For example (see Figure-4), CoCoRaHS has more than 20,000 members while on March 1$^{st}$, 2016 there were only 9,080 reports by 9:30 am. Besides, even though everyone is trained to take the standard rain gauge reading, because it is a volunteer activity, the hands-on measurement can lead to errors.

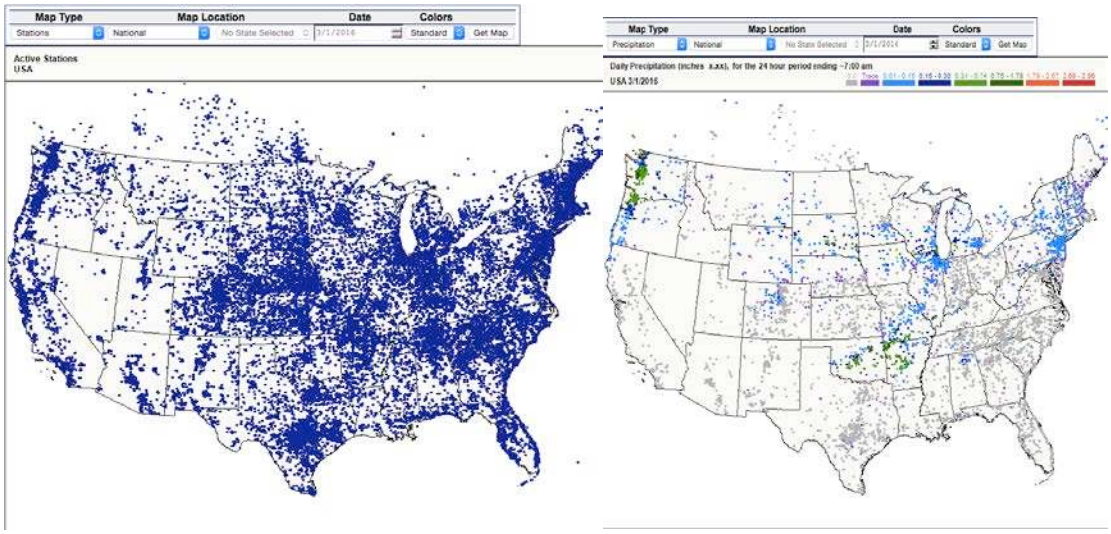



Figure-4 Comparison between the volunteer map and one-day report map

The third restriction is because it is volunteer activity other activity may interfere with the ability to collect daily data. Some people may not have the time or interest to do this hands-on experiment everyday but they still want to know the precipitation in their gardens.

## 1.2 Scope – Can these restrictions be fixed

Despite the three restrictions, CoCoRaHS does a good job providing precipitation information across many parts of the country and some international areas. The question is can the CoCoRaHS rain gauge be modified to eliminate the three restrictions described while keeping the design similar to the current system? There are already 20,000 gauges in the field. It will be a huge waste if the new design had to replace the entire present rain gauge network. Thanks to the detachable construction and modularity of the CoCoRaHS rain gauge, it might be possible to re-design one of the present rain gauge parts.

To minimize the changes of old gauges, the plan is to re-design the bracket only, which will turn the present gauge into a weight-based automated rain gauge. The new design will

also enable the rain gauge to automatically report precipitation data every 15 minutes. This would help realize the near instant precipitation event report and resolve the limits of temporal granularity. Thus, the new design will provide more detailed and dynamic precipitation maps, which could be used for better ground truth data for radar correction.

As a first step towards the development of new CoCoRaHS rain gauge, the contribution of this study is the proof of concept of the low-cost weight-based rain gauge. The research focuses on the Internet of Things (IoT) for a real time rainfall measurement device. The project is divided into two parts, the mechanical design part and communication parts. The mechanical design should provide an accurate low-cost weight-based rain gauge. The communication part will ensure the feasibility of the communication between the rain gauge and the other electrical devices, such as a laptop or cellphone.

## 1.3 Thesis Outline

This thesis is organized as follows. Chapter 2 presents the general design of an automated CoCoRaHS rain gauge, the description of device components, and bill of material. Chapters 3 and 4 present the detailed design and experiments. Chapter 3 covers the mechanical design based on the present CoCoRaHS rain gauge and testing experiments for the accuracy of the weight-based rain gauge. Chapter 4 describes electronic design, including coding, and communication through Wi-Fi. The thesis is concluded in Chapter 5 with a discussion of contributions of the thesis and future work.

# CHAPTER 2

# DESIGN SCHEME

## 2.1 General

The purpose of the new design is to turn the present CoCoRaHS rain gauge into a weight-based automated rain gauge. The new rain gauge could not only be manually read for the height of precipitation by volunteers, but also sense the weight of precipitation and then convert this weight into the height of precipitation automatically. Volunteers could continue their tradition to report the daily reading but the new rain gauge could report the intermediate time data in a rainfall event automatically. As will be shown, it is expected that the total cost of new rain gauge could be less than fifty dollars for current customers and seventy-five dollars for new customers.

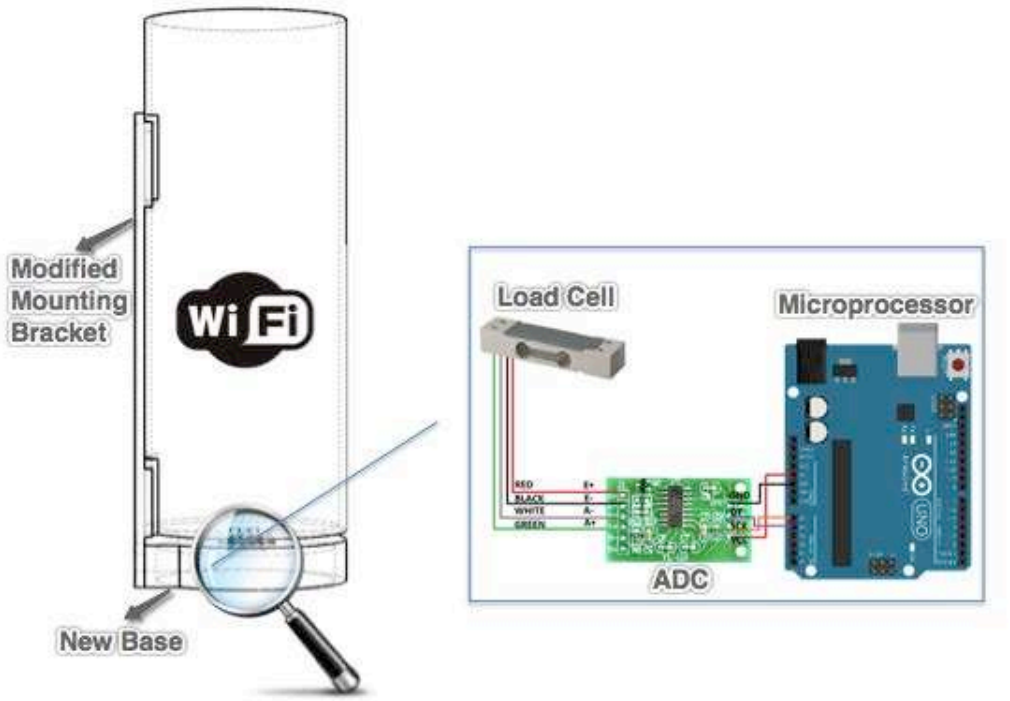



Figure-5 Design of new CoCoRaHS rain gauge

## 2.2 Components

The design of the new rain gauge consists of five parts, including base bracket, load cell, analogue digital converter (ADC), microprocessor, and Wi-Fi communication subsystem. The load cell will be used to measure the weight of precipitation. Because of the analogue signal coming out of the load cell, an ADC module is used to amplify the signal and convert it to digital data. Once the data is in digital form, the microprocessor program will convert the weight of precipitation to the height of precipitation. The Wi-Fi will be the main method of communication between the new rain gauge and other electronic devices. All of the datasheets can be found in Appendix A.

### 2.2.1 Load cell

To measure the weight of the precipitation in the rain gauge, a sensor is required with enough capacity to support the weight of the rain gauge cylinder and its full precipitation capacity and accurate enough to detect small weight changes. The weight of the rain gauge will vary from 0.7kg to 2.9kg. The empty weight of rain gauge is 0.7kg and its capacity is 11inches of precipitation, which is 2.2kg of water. The resolution of the present CoCoRaHS rain gauge is 0.01 inch, which is equal to 2g as weight. Therefore, the weight sensor needs to have a capacity of 3kg and a resolution less than 2g. This range of weight-based measurements is common to jewelry and small kitchen scales. Therefore, a load cell from those scales could provide the function needed in the new CoCoRaHS rain gauge.

According to the datasheet of 3133 - Micro Load Cell (0-5kg) [10], a load cell is a force-sensing module. This module is carefully designed in metal structure, with small elements,

known as strain gauges, mounted in precise locations on the structure (see Figure –6). Load cells are designed to measure force in one direction and ignore other forces being applied. Normally for the best result, it is suggested to at most use 1/3 to 1/2 of the measuring range of the load cell to prevent the plastic distortion in strain gauges. Therefore, in the design for the CoCoRaHS rain gauge, the normal operation of CoCoRaHS rain gauge is less then 1kg, which requires a 5kg load cell to avoid overload.

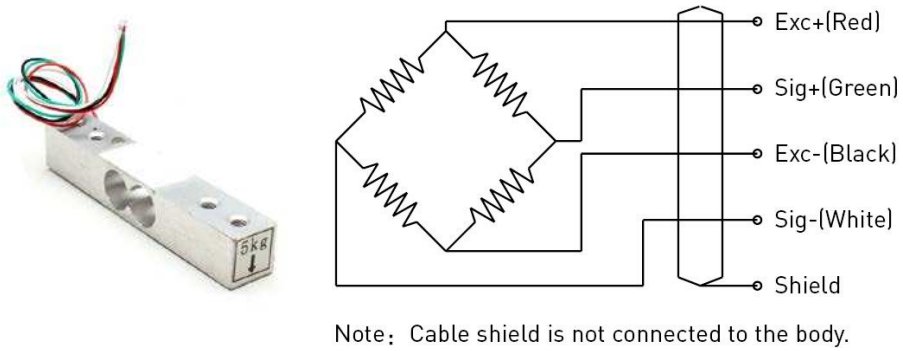



Figure-6 Bridge Circuit of Load Cell [11]

Strain-gauge load cells convert the load acting on them into electrical signals. The measuring is done with very small resistor patterns, also known as strain gauges. It is an effectively small, flexible circuit board. From the description in the Datasheet, “the gauges are bonded onto a beam or structural member that deforms when weight is applied, in turn deforming the strain gauge. While the strain gauge is deformed, its electrical resistance changes in proportion to the load.” [10] The main problem is the sensitivity to temperature variation. This circuit is highly influenced by temperature. To maintain the accuracy of force measurement, there are two methods to cancel out the effects of temperature. The first is by matching the expansion rate of the strain gauge to the expansion rate of the metal it is mounted on; undue strain on the gauges can be avoided as the load cell warms up and cools

down. The other method is using multiple strain gauges to solve the temperature compensation. The theory of this method is that all strain gauges respond to the change in temperature with the same change in resistance. “Some load cell designs use gauges that are never subjected to any force, but only serve to counterbalance the temperature effects on the gauges that measure force. Most designs use four strain gauges, some in compression, some under tension, which maximizes the sensitivity of the load cell, and automatically cancels the effect of temperature.” [10] The load cell chosen belongs to the second method and is know as a strain gauge bridge sensor.

2.2.2 Analogue digital convertor

The electrical signal output by the load cell is very small and cannot be read directly by the microprocessor. The original signal requires specialized amplification and then coverts the analogue signal to digital data. Therefore, a module is needed to perform the amplification and conversion of the load cell output. Based on Avia Semiconductor’s patented technology, HX711 is a precision 24-bit analog- to-digital converter (ADC). It is designed for weight scales and industrial control applications to interface directly with a strain gauge bridge sensor.

According to the HX711 datasheet, there is a two input-multiplexer named Channel A or B. These two Channels are differential inputs to the low-noise programmable gain amplifier (PGA) [12]. “Channel A can be programmed with a gain of 128 or 64, corresponding to a full-scale differential input voltage of ±20mV or ±40mV respectively, when a 5V supply is connected to AVDD analog power supply pin. Channel B has a fixed gain of 32.” [12] There

is an on-chip power supply regulator, which eliminates the need for an external supply regulator to provide analog power for the ADC and the sensor. “Clock input is flexible. It can be from an external clock source, a crystal, or the on-chip oscillator that does not require any external components. On-chip power-on-reset circuitry simplifies digital interface initialization. There is no programming needed for the internal registers. All controls to the HX711 are through the pins.” [12]

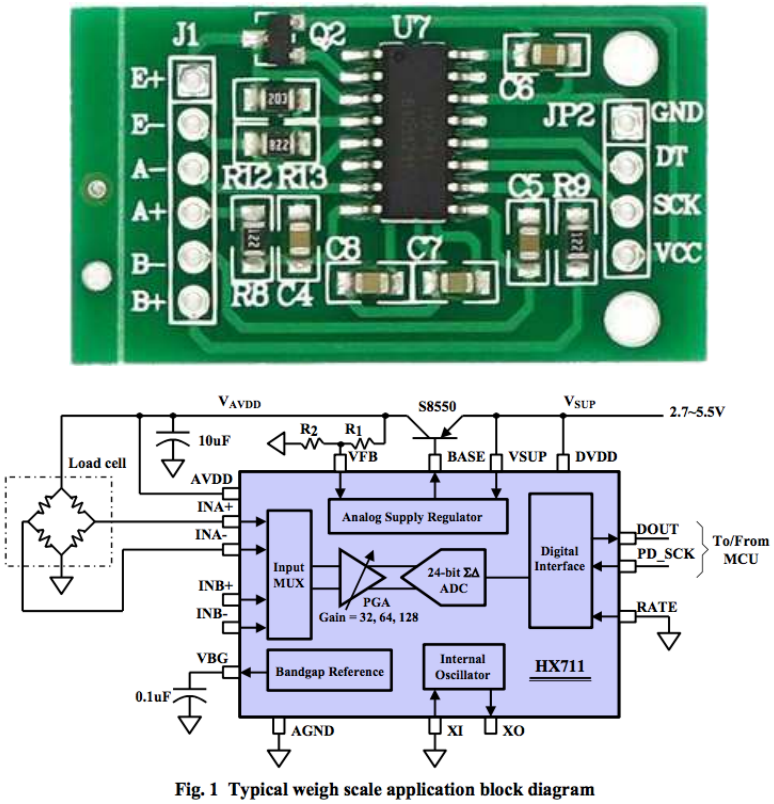



Figure-7 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales [12]

2.2.3 Microprocessor

The Arduino Uno R3 microprocessor was selected as the controlling microprocessor. According to Arduino website [13], the Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins, 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything

needed to support the microcontroller; it can be connected to another computer with a USB cable or can be independently powered by an AC-to-DC adapter or battery.

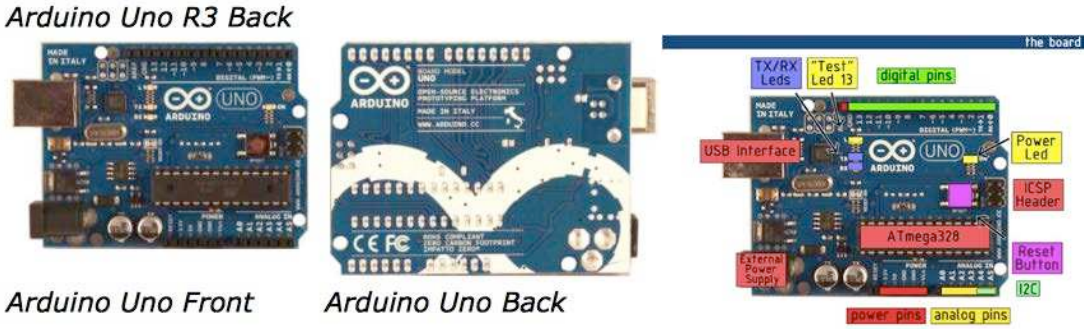



Figure -8 Arduino Uno [13]

2.2.4 Wi-Fi

Wi-Fi networking was selected for the rain gauge. And EMW3162 for quick development cycles was chosen because of its interface as platform to transmit wireless data.

As the EMW3162 datasheet states [14], “EMW3162 is a low-power embedded Wi-Fi module that integrates a wireless LAN MAC/baseband/radio, and a Cortex-M3 microcontroller STM32F205 that runs a unique ‘self-hosted’ Wi-Fi networking library and software application stack. The EMW3162 has 1M bytes flash memory, 128k RAM and rich peripherals for your embedded Wi-Fi applications. “ EMW3162 is pin compatible and designed to directly interface with the Arduino.

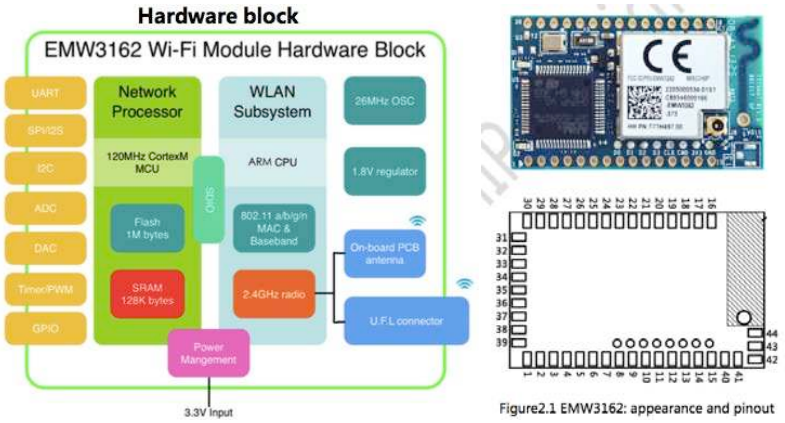



Figure -9 EMW3162 [14]

2.2.5 Mechanical system (Bracket)

The Single Point Load Cell selected needs to be mounted by bolting down the end of the load cell where the wires are attached and applying force on the other end in the direction of the arrow shown in Figure-10. According to load cell handbook [12], “where the force is applied is not critical, as this load cell measures a shearing effect on the beam, not the bending of the beam. If you mount a small platform on the load cell, as would be done in a small scale, this load cell provides accurate readings regardless of the position of the load on the platform.” Therefore, a base is needed under the rain gauge cylinder for the location of the load cell.

To maintain most of components of the original CoCoRaHS rain gauge design requires changes to only be made to the bracket. The load cell requires a base to be added to the bracket that could include all the electronics fitted within this base. The original CoCoRaHS is made from plastic, which matches to the use of 3D printing technology for the construction of the base.


Figure -10 5kg load cell with the sign of arrow

## 2.3 System lifetime

Under normal usage, the redesigned CoCoRaHS rain gauge should offer a minimum six-month lifetime on battery power. In order to minimize the power consumptions, the system should be operated at the minimum required voltage. For the components selected this is 3 volts. Therefore, one possibility for a battery pack would be to use two lithium-polymer 3.4-volt AA size batteries. These batteries are rated at 2300mA-hr. Analyzing the power consumed by the system, the main concern will be the radio which consumes 100 times more energy than the other electronic components. The lifetime of the rain gauge battery will be strongly influenced by geography because different rainfall patterns affect the frequency of the radio wake-up. Both of the extreme situations can be calculated, no precipitation and continuous precipitation. For the no precipitation situation, the upper-limit of the battery life is about fifty years. On the other hand, if the radio operates continuously to transmit data, the lower-limit of the battery is less than one day, i.e. about fifteen hours. So the battery pack could power this device between 15 hours to 50 years.

A more realistic data report would be every 15 minutes and at least once per day. Calculating on these more realistic conditions, we get the rain gauge lifetime from two years to 50 years. Therefore, these components satisfy the requirement for the rain gauge battery lifetime.

| Part | Mode | Current |
|---|---|---|
| load cell | sleep | 0μA |
| | wake-up | N/A |
| HX711 | sleep | 0.3μA |
| | wake-up | 1400μA |
| AVR | sleep | 0.9μA |
| | wake-up | 1200μA |
| Wi-Fi | Off | 2μA |
| | wake-up | 270000μA |

## 2.4 Bill of Material

Because this is just a prototype, the cost of the final product will be much cheaper than what was spent on the proof-of-concept. The following chart presents the cost spent for the prototype while the 'expected wholesale price' means the price when the new rain gauge is manufactured in quantity. There are additional costs that cannot be predicted such as the design fee for integrating all the electronic components onto a specially designed printed circuit board and the mold design and construction and set-up costs of the new bracket and base production.

| Item # | Qty | Mfg Part | Retail Price | Expected Wholesale Price |
|---|---|---|---|---|
| 1 | 1 | 5kg load cell | $5.52 | $2 |
| 2 | 1 | HX711 module | $12.52 | Combine these three components into one PC board, estimated $25 |
| 3 | 1 | Arduino | $16.99 | |
| 4 | 1 | Wifi module | $9.95 | |
| 5 | 2 | Battery | $1 | $0.3 |
| 6 | 1 | Plastic parts | N/A | N/A |
| 7 | | Others | | |
| | | Total | $50+Plastic | $27.50+Plastic |

# CHAPTER 3

# MECHANICAL DESIGN AND TESTS

## 3.1 General

In this chapter the mechanical system of the rain gauge is discussed and the experiments with the load cell are shown. The mechanical system includes a funnel, a measuring tube, a 4-inch diameter overflow tube, and a mounting bracket (see Figure-11). To maintain as much of the present CoCoRaHS rain gauge as possible, the decision was made to only modify the mounting bracket and to design a new base for the electronic components (see Figure-12). The experiments with load cell were to measure the linearity between the weight input and analogue output from the load cell.

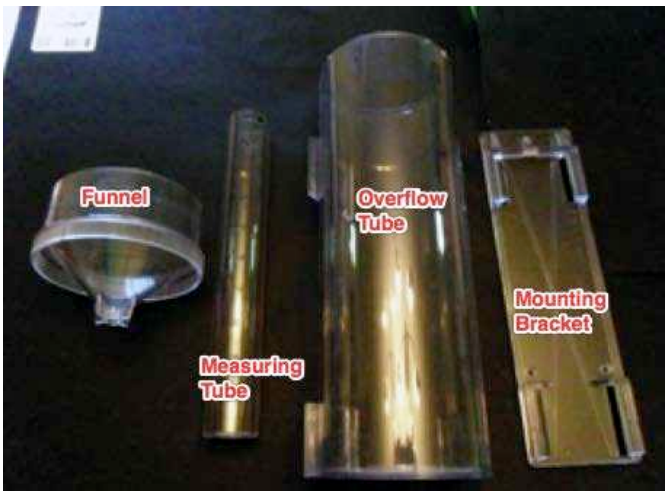



Figure-11 components of present CoCoRaHS rain gauge

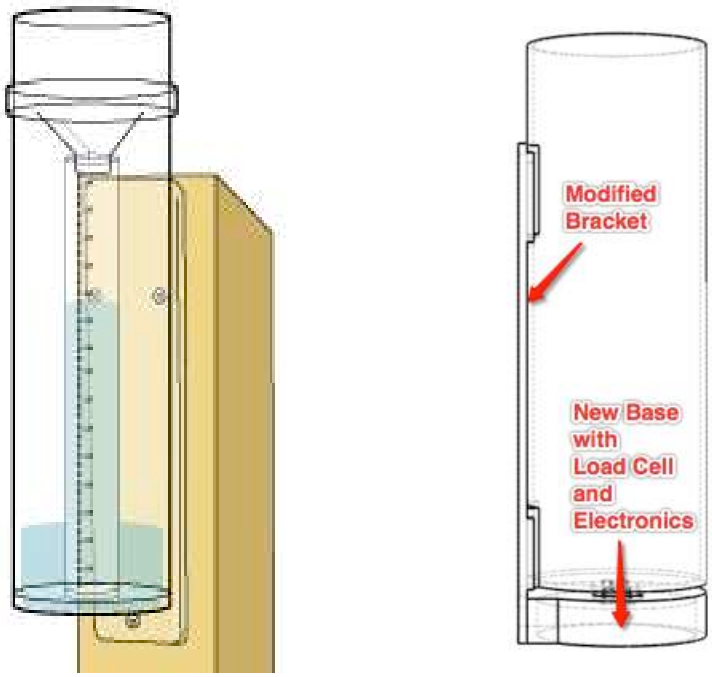



Figure 12 Rain gauge models

(left) present rain gauge

(right) new rain gauge

3.2 Bracket modification

The present mounting bracket has stops that hold the cylinder against the bracket (see Figure-13 left). Using a weight-based method requires that the cylinder needs to be free to move in the vertical direction (see Figure-14). Therefore, to minimize changes to the original design of the bracket, the stops on the original bracket were removed to enable the rain gauge cylinder to slide through the bracket. The material of the bracket is polycarbonate, thus the removal method used a Dremel Tool as a grinder to limit damage to the bracket. The detail of altered bracket is shown in Figure-15.

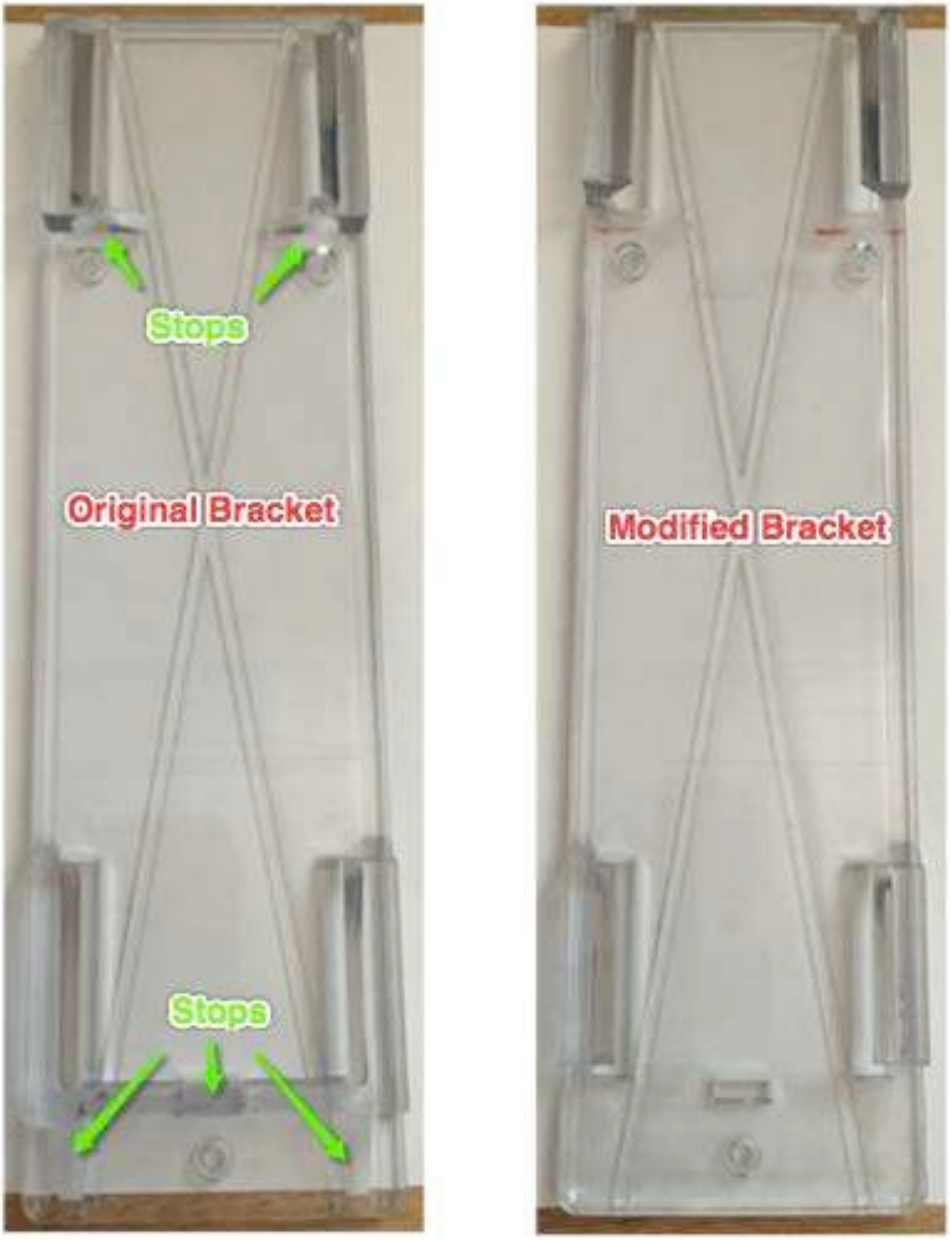



Figure – 13  the bracket

(left) original bracket with stops;

(right) modified bracket without stops

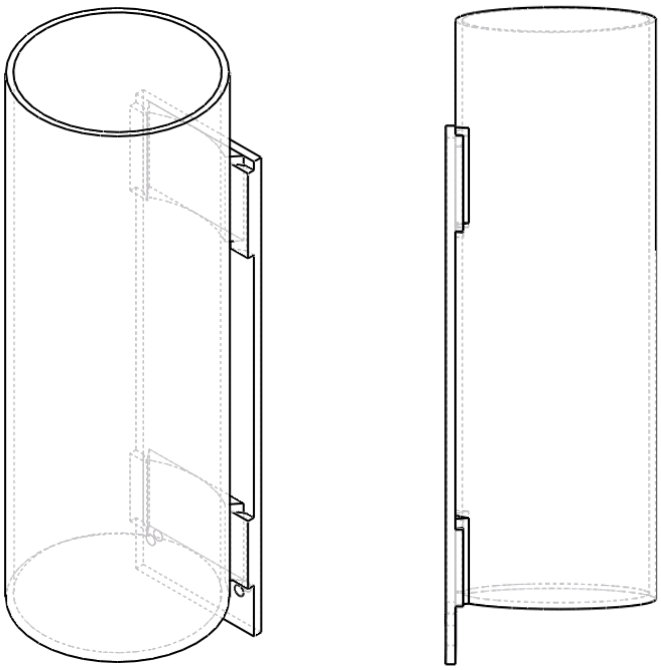


Figure –14 the new bracket

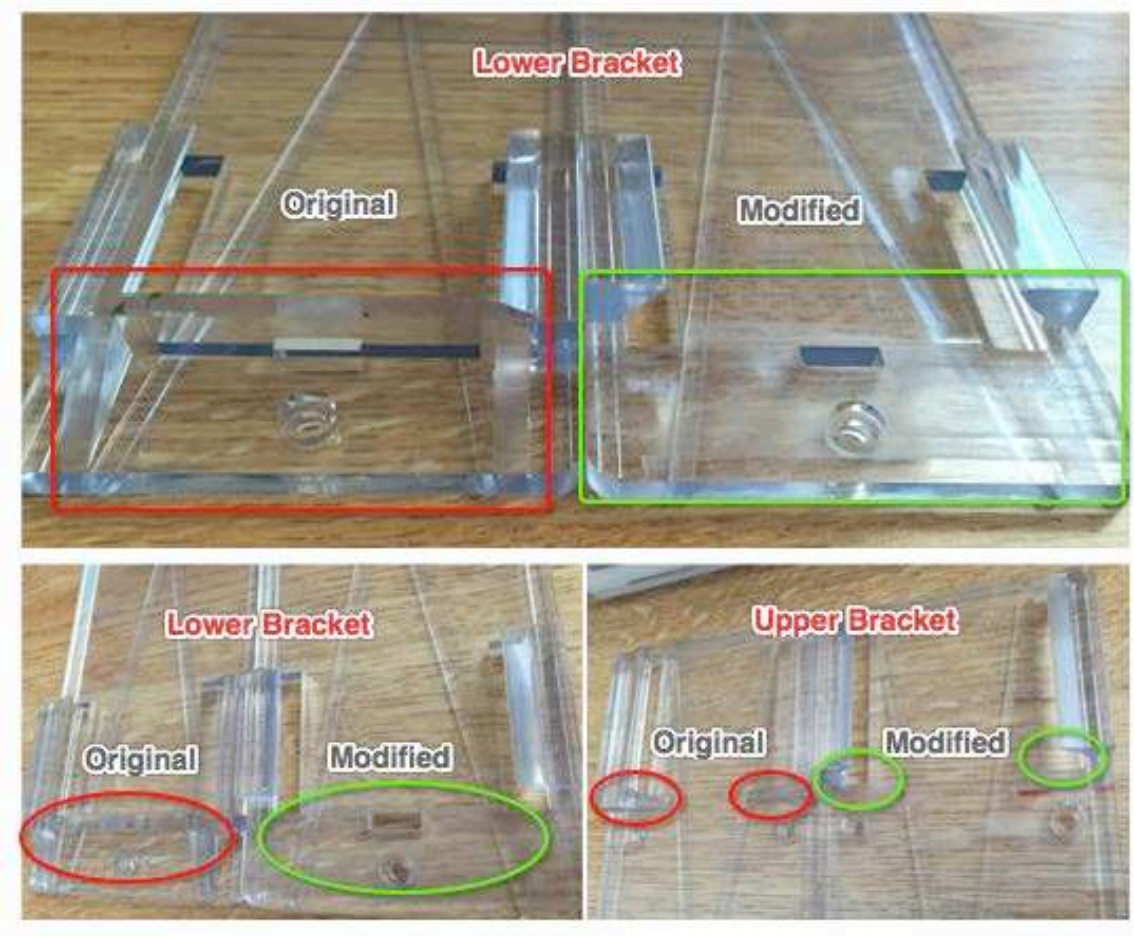



Figure – 15 the detailed comparison of the brackets

### 3.3 New base design

To support the load cell and contain the other electrical parts, a base is needed that could match with the bracket (see Figure-16). The shape of this base is a circular on a rectangle that is easy to mount on the bracket and interface with the cylinder. The base will house the single point load cell and the electronics, batteries and communication system. The 0-2kg load cell used for the new base design was removed from a kitchen scale. Although the 0-2kg load cell does not satisfy the actual requirement of the CoCoRaHS rain gauge, it still could be used in the proof-of-concept. This piece screws to the bracket for a firm connection. 3D-printing technology was used to build the base.

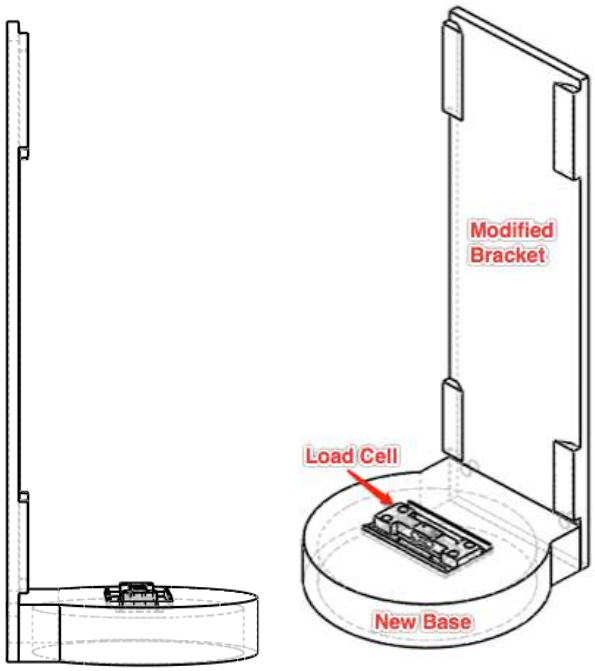



Figure – 16 the propose model for the base

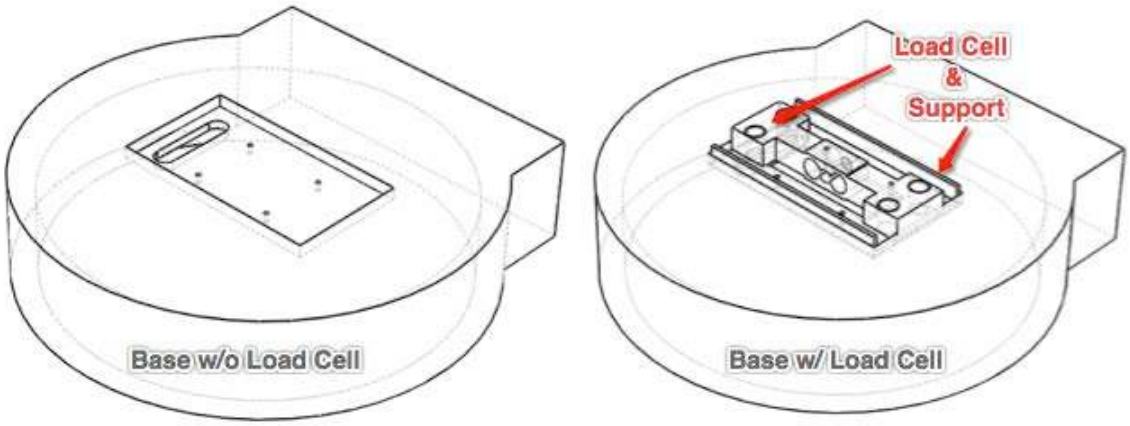



Figure –17 Base Design

(left) base without load cell

(right) base with load cell support and load cell

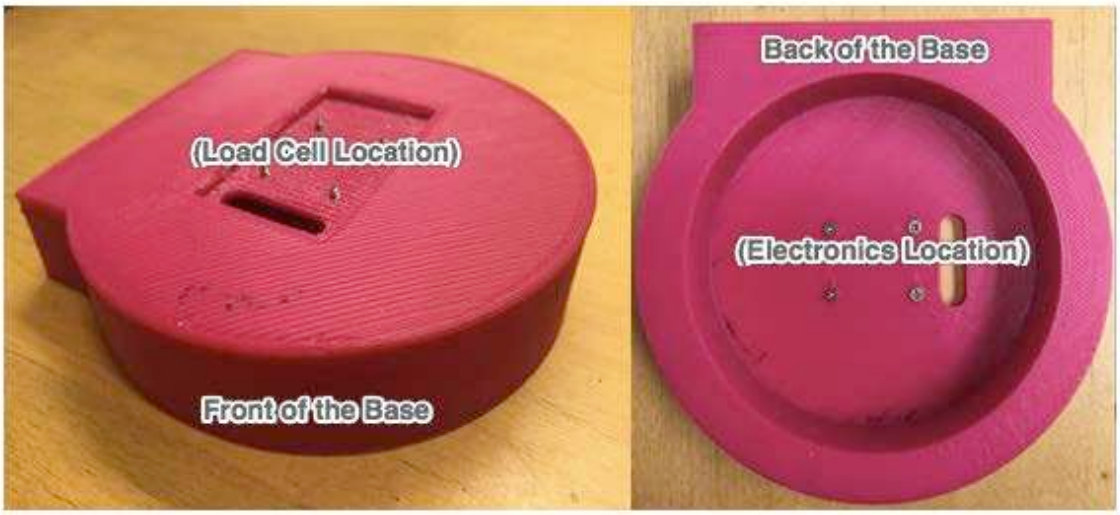



Figure – 18 the 3D printed base

3.4 Experiments with load cell

There were two kinds of experiment to demonstrate the feasibility of this weight-based rain gauge. The first set of experiments was to test the linearity of load cell output. The other set of experiments was to see how well the base works with the bracket.

### 3.4.1 Linearity experiments

The load cell was tested under different voltage inputs (see example in Figure-19). The suggested voltage input should be 5 volts, which will consume more power than lower voltages. There was concern that the linearity would not be as good under a lower voltage input. The experiments show there is a difference in the slope but that the results are linear. This showed that the load cell could be run at low-level voltage input.

To satisfy the requirement of rain gauge lifetime, 3 volts is used as input for testing the linearity of the two load cells. The experiment equipment includes a power supply, a digital voltmeter, a plastic container/rain gauge cylinder, and the load cell attached to the base. The measurement devices are located in C105 Lab where no liquid is allowed in the Lab. Therefore, rice was used as weight rather than water. Because of different shape of the load cells, different bases were applied to install different load cells. The experimental set-ups are shown in Figure-20.

The actual useful range of load cell is from 0g to 1kg because it is rare to have a filled CoCoRaHS rain gauge in the field. The 2kg load cell from the kitchen scale was chosen to test the linearity under different increments. If the input weight is assumed as ‘g’ and the output voltage as ‘V’, the equation of linear regression for each increment can be calculated. These equations are also shown in Figure-21 ‘Function comparison’. All the functions have similar slopes. The results shown in Figure-21 demonstrated that the load cell had enough sensitivity and is linearly stable.

To consider the extreme situation that CoCoRaHS rain gauge is filled with water, the 0-5kg load cell should be used to test the linearity over the maximum range of the the rain gauge. Because the size of most 0-5kg load cells in market is bigger than the 0-2kg load cell from the kitchen scale, the two cells were unable to share the base. A new metal base (shown in Figure-22) was designed and built by the CSU machine shop for testing. The results (shown in Figure-23) again demonstrated the high linearity of the load cell output, which re-confirmed the feasibility of the weight-based method.

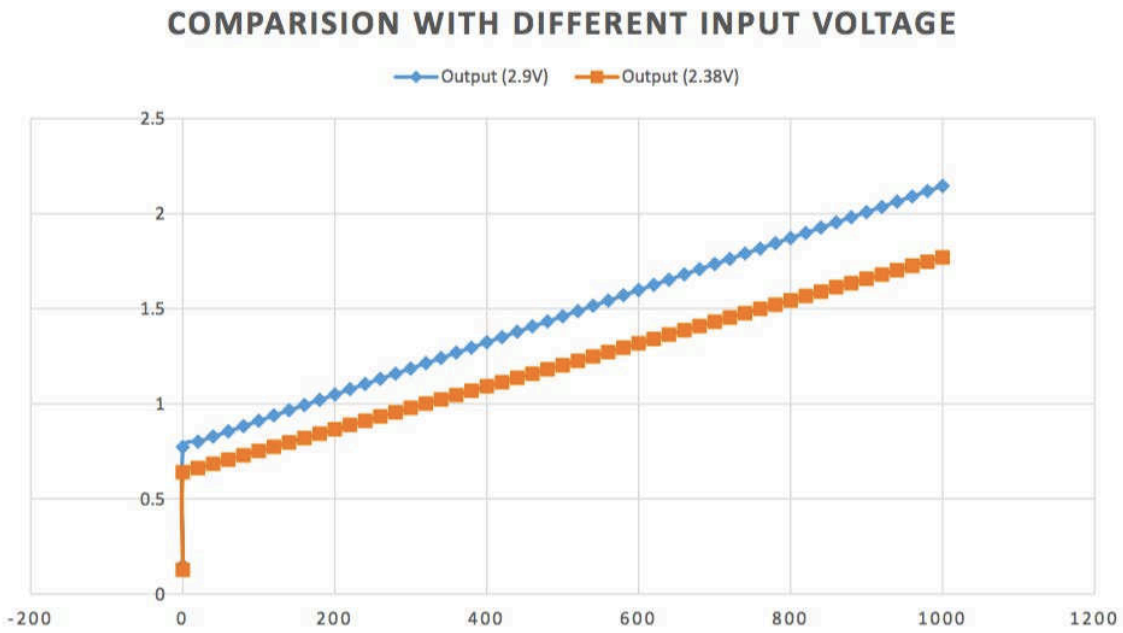



Figure – 19 Linearity with different voltage input

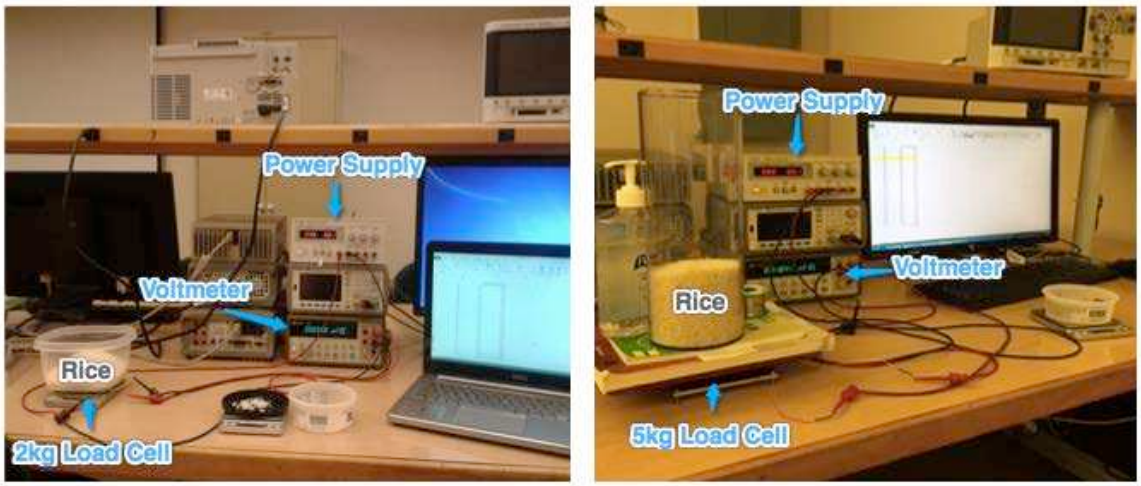
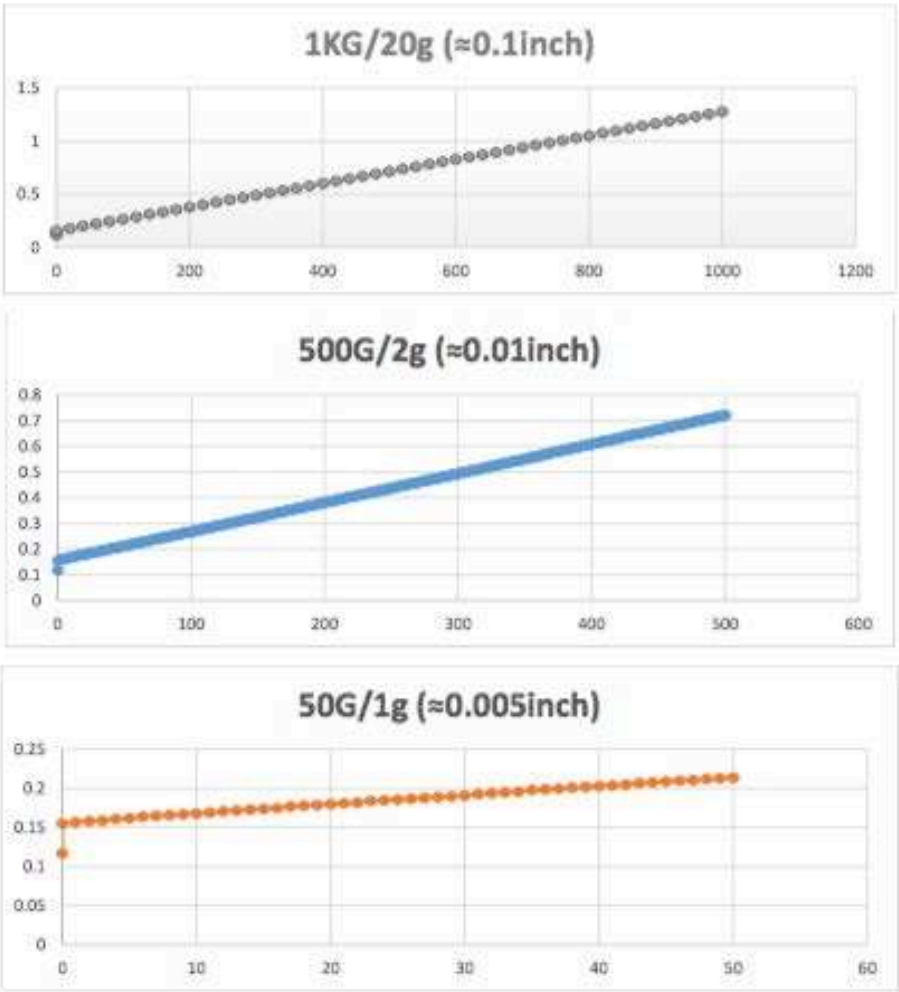



Figure – 20 Linearity experiments set-up

(left) 2kg load cell

(right) 5kg load cell

***Chart Comparison:***




***Function Comparison:***

1KG/20g (≈0.1inch): V (g) = 0.001121041g+0.153558069

500G/2g (≈0.01inch): V (g) = 0.001130392g+ 0.155405

50G/1g (≈0.005inch): V (g) = 0.001168416g+ 0.1564170

Figure – 21 Results and analysis of linearity experiments

(x-axis represents input weight, g; y-axis represents output voltage, mV).

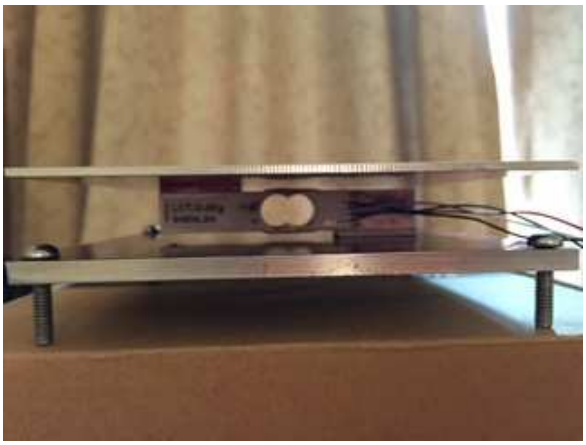


Figure-22 Metal platform and 5kg load cell

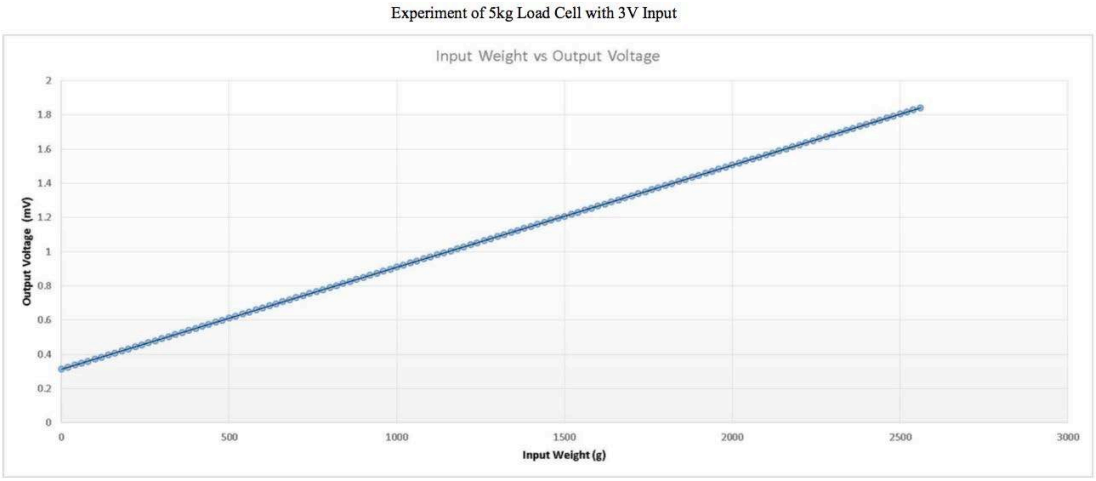



Figure-23 The result of 5kg load cell

### 3.4.2 Base-bracket experiments

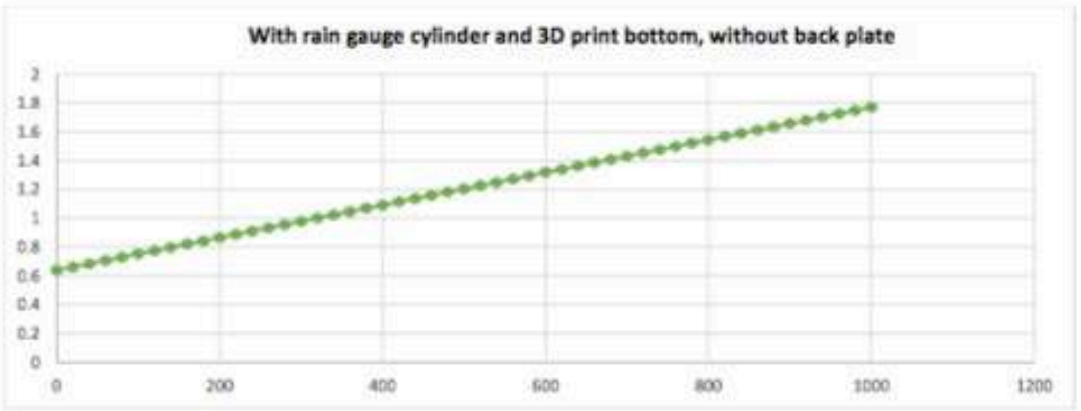
Based on the linearity experiments, the feasibility of the weight-based method was demonstrated. After the new base was designed and 3D-printed, the following set of experiments focused on the interaction between the new base and the bracket. The new base needed to replace the kitchen scale base for the load cell and the bracket was stable with the new base. Simulating the actual situation as much as possible, the rain gauge cylinder was

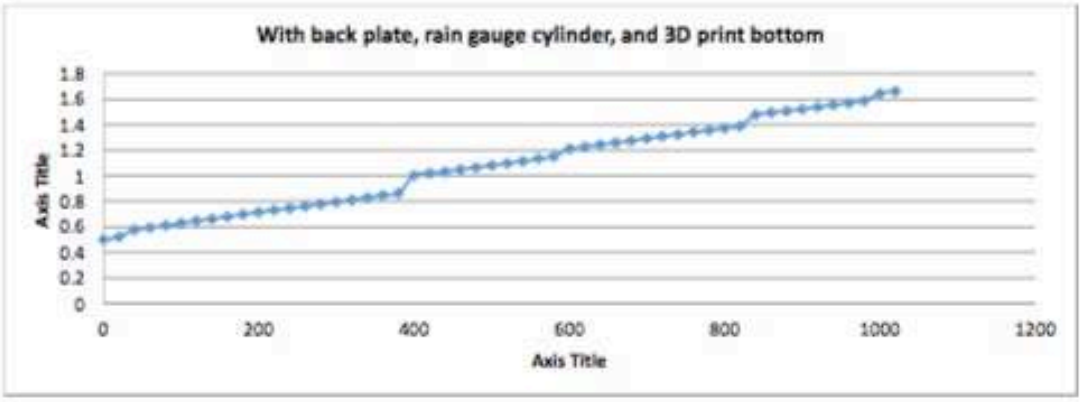
used as the container. Two experiments were designed to compare with the results in the linearity experiment, which was assumed as ideal situation.

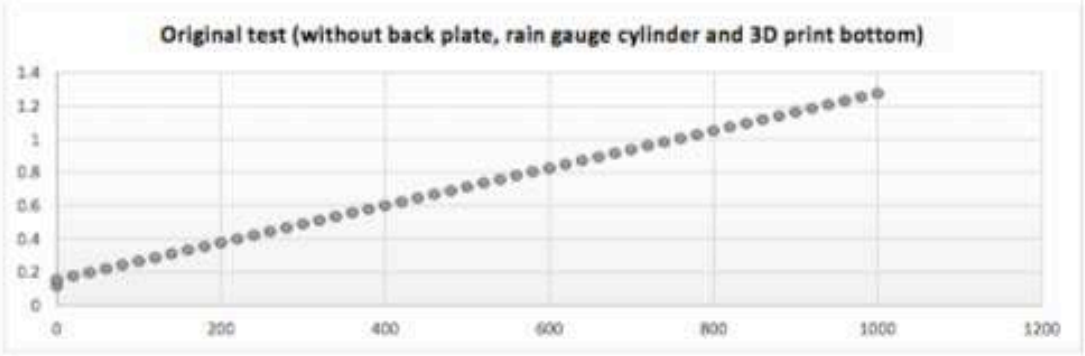
One of the experiments used only the rain gauge cylinder and the 3D-printed base but without the bracket. The purpose of this experiment was to test the 3D-printed base. The graph of the results is shown in Figure-24 in green, which shows high linearity and continuity.

The other experiment is with the rain gauge cylinder, 3D-printed base and the bracket. This experiment was designed to show if the bracket and the base could work together to provide linear results. The graph of the results is shown in Figure-24 in blue. In general, it is linear but with some unexpected points that have permanently influenced the results. The data between the two adjacent unexpected points, which is considered as one interval, is linear and continuous. Each interval has the same slope as the other. It is speculated that the bracket causes this phenomenon. When the weight of the cylinder becomes heavier, the plastic bracket at first tolerates the force then over a certain weight the slight bending of the bracket permanently influences the load cell reading.

The comparison of charts and equations is shown in Figure-24. The conclusion from these experiments was to improve the connection between the base and the bracket. The better the connection between base and bracket, the smaller the influence the bracket has on the results. Better design possibilities will be described in Chapter 5 as suggestions for future product designs.




Assume output voltage V is the function of increasing weight g:

| | |
|---|---|
| **With rain gauge cylinder and 3D print bottom, without back plate:** | V (g) = 0.00113143g+ 0.638395918 |
| **With back plate, rain gauge cylinder, and 3D print bottom:** | V (g) = 0.00113162Sg+ 0.497544267 |
| **Without back plate, rain gauge cylinder and 3D print bottom:** | V (g) = 0.001121041g+ 0.153558069 |

Figure – 24 Results and analysis of Cooperation experiments

(x-axis represents input weight, g; y-axis represents output voltage, mV).

# CHAPTER 4

# ELECTRONIC DESIGN AND TESTS

## 4.1 General

To minimize energy consumption, this new CoCoRaHS rain gauge will have two operating modes, sleep and wake-up (see in Figure-25). Sleep mode is the normal state where the rain gauge operates at a very low level of power. Wake-up mode has three options depending on the time and the precipitation. Every fifteen minutes the system wakes up to measure the weight of the cylinder (see in Figure-26). If there is a weight change, the microcontroller will wake up the Wi-Fi to transmit the precipitation data. If the weight has not changed, the whole system will go back into the sleep mode. Everyday at a specific time (like 7 a.m.), the Wi-Fi unit is activated to send the daily report.

This chapter mainly describes the coding for the microcontroller and the experiments. The experiments have two parts. One is to test the operation of the electrical system after connecting all components. The other is to demonstrate the wireless transmission between rain gauge and a receiving laptop.



Figure-25 Two basic modes

Figure-26 Three options for wake-up mode

## 4.2 Connection setup

As shown in Figure-27, the strain gauge bridge is connected to the HX711 analog inputs and HX711outputs to the microcontroller board. This system allows for a load cell with any weight range. In these communication tests the 0-5kg load cell was used as meeting the requirements of the CoCoRaHS rain gauge.

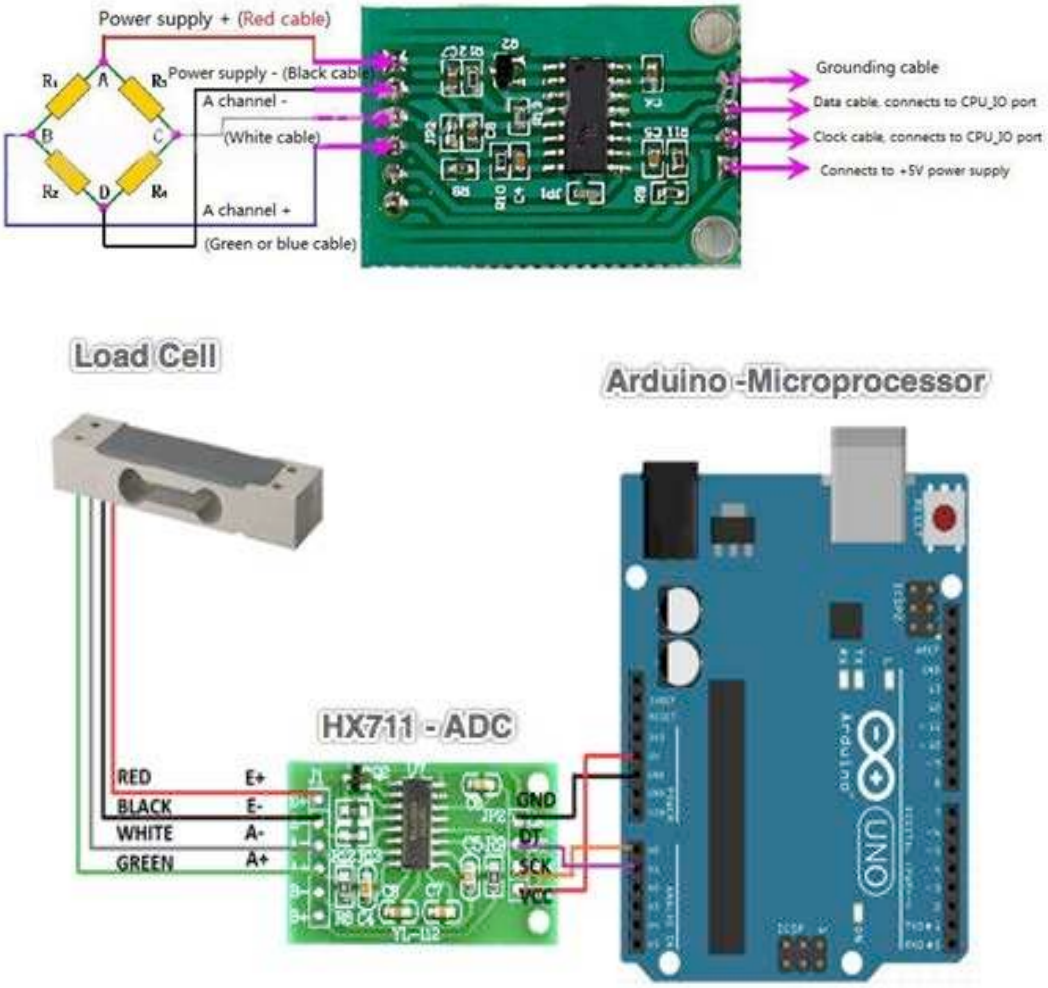



Figure-27 Electrical Connection

a) Bridge circuit connection (http://rohmadi.my.id/wp-content/uploads/2014/10/HX711-3.png)

b) Material connection [15]

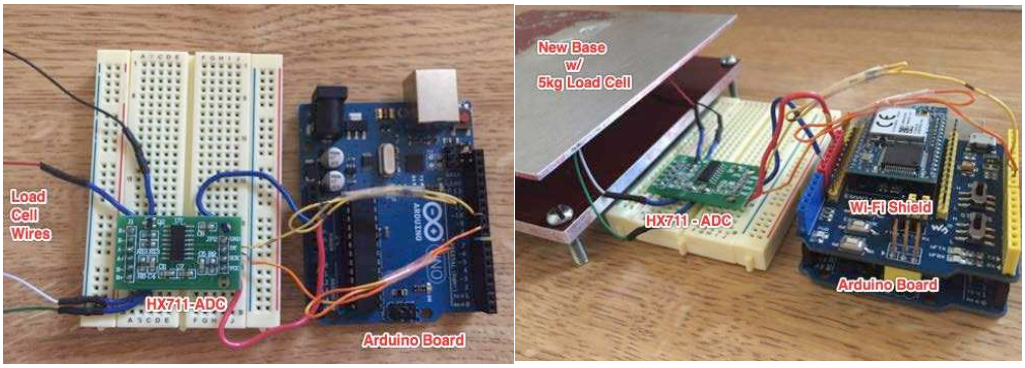



Figure-28 Setup

There are several kinds of external power sources for Arduino. Arduino UNO has four power input methods (see Figure-29). The first is through a USB port, which should be +5V input. The second is a JAPAN JACK socket, which accepts from +7V to 12V. The third is Vin socket and the last is a 5V socket. In these experiments, the system is powered from the laptop through the USB port to Arduino. The USB power was the easiest and most convenient compared to the other methods.

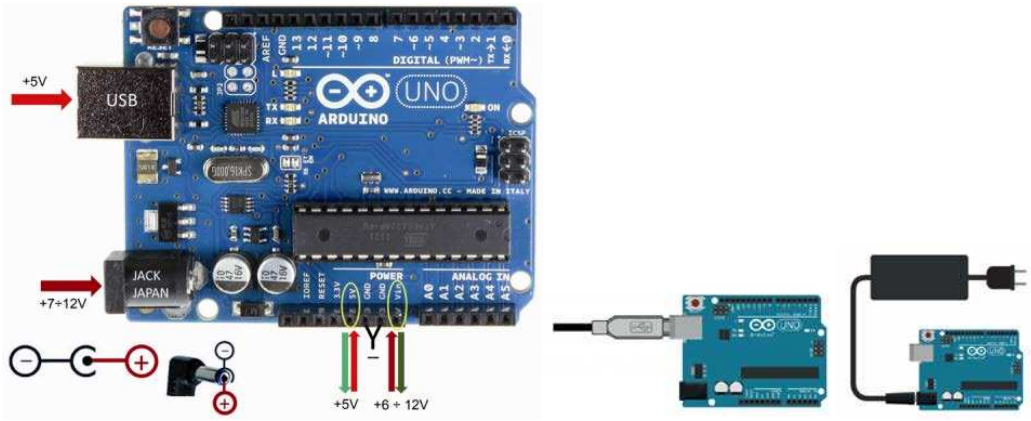



Figure-29 the inputs for Arduino powering [16]

## 4.3 Coding of Arduino

The coding for the Arduino is vital to the whole system, controlling unit conversion, basic calculations, data storage, and wake-up control. For the proof-of-concept the code does not need to satisfy all the desired functions. The main goal for the code is to obtain and output a

value with weight units. There is an Arduino library for the HX711, which is simple to use. Only one function is called to get the reading from the chip. The detail of HX711 library is shown in Appendix-B.

Before everything starts, the unit is calibrated with two known weights. One is to assume the platform without load as 0g. The second is any known weight, such as 100g standard weights. Use the sample code of serial reading (see in Figure-30). There are two numbers, 'offset' and 'coefficient'; that are needed to do unit conversion. When there is no load, the direct reading is 'offset'. When it is loaded with 100g, the direct reading is 'X'. The follow calculation produces the coefficient for unit conversion.

$$coefficient = \frac{100g - 0g}{(offset - X)}$$

After 'offset' and 'coefficient' are obtained, the code for the load cell is complete. The code is shown in Figure-31.

```
/*Sample of Serial reading from HX711 Library*/
#include <HX711.h>                    // call HX711 library

HX711 hx(9, 10);                      // HX711 Function (DOUT, SCK)
void setup() {


void setup() {
  Serial.begin(9600);
}

void loop()
{
  double sum = 0;
  for (int i = 0; i <= 20; i++)       // loops 20 times
    sum += hx.read();                 // accumulation
    Serial.println(sum/20);           // averaging, display the direct reading

}
```

Figure-30 Sample code for HX711

```
#include <HX711.h>                    // call HX711 library
HX711 hx(9, 10, 128, 0.002281304);    // HX711 Function (DOUT, SCK, amplifier
                                      // gain, coefficient from calibration)

void setup() {
  Serial.begin(115200);
  hx.set_offset(241800);              // this value is obtained by calibrating the
                                      // scale with known weights

}
void loop() {
  delay(500);                         // waits for half second
                                      /* to reduce error, the display reading is
                                      the average 20 times readings*/

  double sum0 = 0;
  double sum1 = 0;
  double a = 0;
  for (int i = 0; i <=20; i++) {      // loops 20 times
    sum0 += hx.read();                // accumulation of direct reading from
                                      // hx711
    sum1 += hx.bias_read();           // accumulation of the reading with
                                      // correction factor
  }
  Serial.print(sum0/20);              // averaging, display the direct reading
     Serial.print(char(9));           // display tabulator
  Serial.print("       weight =  ");  // display chart "weight="
    Serial.print(char(9));            // display tabulator
  Serial.println(sum1/20);            // averaging, display the weight
}
```

Figure-31 5kg load cell code

## 4.4 Experiments with Arduino and Wi-Fi shield

This set of experiments has two parts. The first part runs the script to test the code. The second part executes the wireless data transmission. The only difference between these two parts is the use of Wi-Fi module. These experiments are shown in Figure-32. Water is used as the weight. The water was added by syringe in increments of 5 ml to ensure incremental consistency.

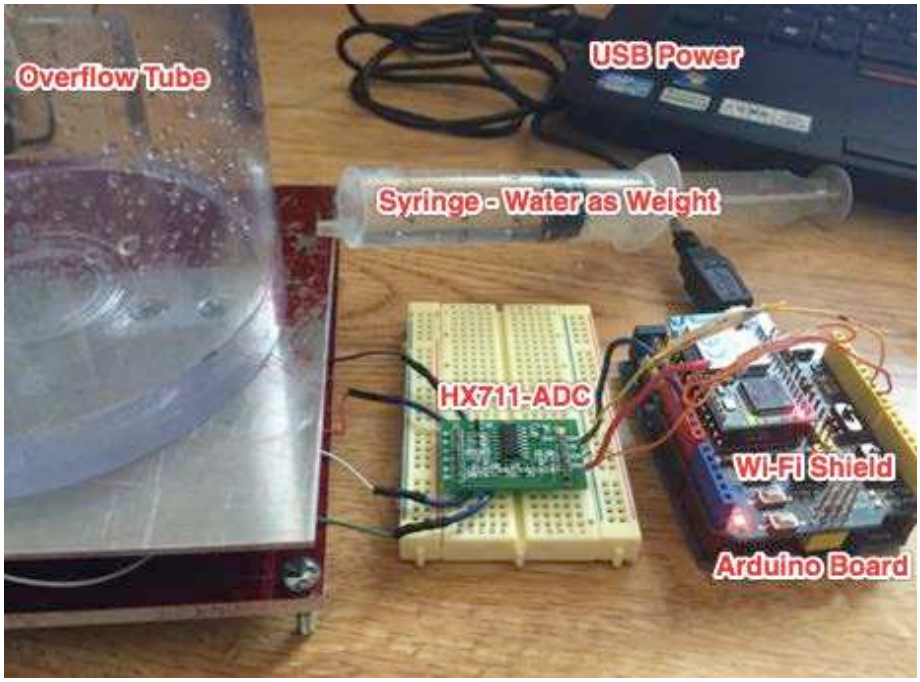



Figure-32 Setup

The code testing experiments use Arduino Software (IDE) to program and display results on a serial monitor (see Figure-33). The first step was to run the sample code of HX711 to get ‘offset’ and ‘coefficient’ numbers. These two numbers were manually input into the code for 0-5kg load cell. After the Arduino was programmed properly, opening the serial monitor allowed the display of the real-time data.

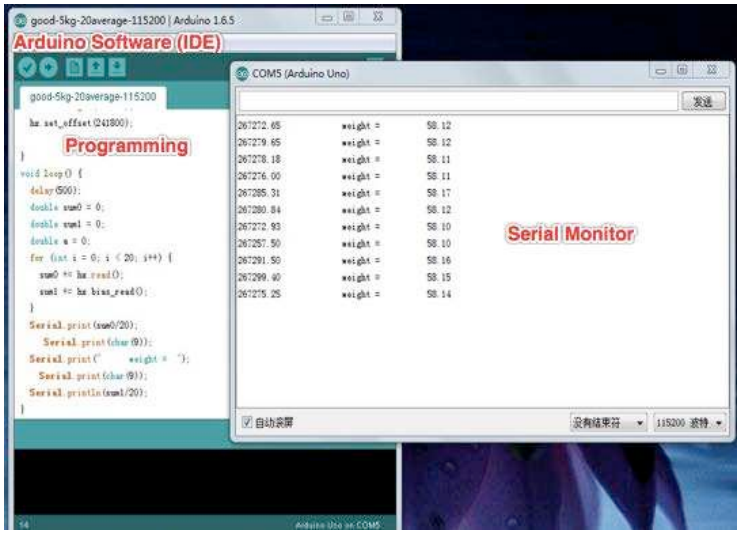



Figure-33 IDE and Serial Monitor of Arduino

The wireless transmission experiments mainly test the operation of the Wi-Fi module and the receiving network. The Arduino Wi-Fi shield was plugged into the Arduio UNO board (see Figure-34). A laptop or a tablet was used as the receiving device to simulate the network. The receiving device needs to have ‘Netassist’ installed to display the data (see Figure-35 left). The wireless transmission was as fast as getting data from the USB port.

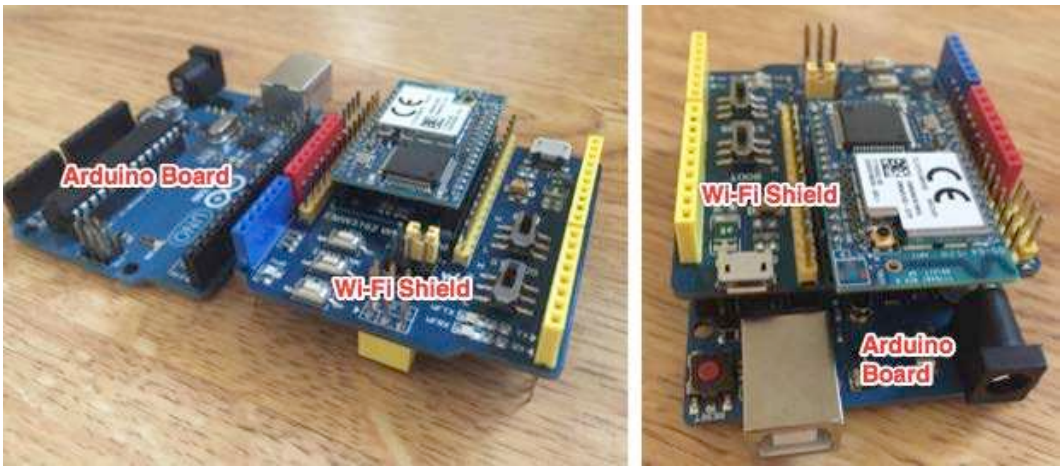



Figure-34 Wi-Fi shield for Arduino


Figure-35 Data display comparison

(left) Netassist

(right) Arduino serial monitor

## 4.5 Experiments of data stability

Based on the above experiments, the display of certain weights is floating, not static. Two more experiments were designed to record the floating data and try to determine whether there is a pattern. One experiment tests the range from 0g to 50g with 2g as increment (see in Figure-36). The measurement for each weight is collected with 2 minutes and 30 seconds, which is about 17~18 data points for each measurement. The range of the other experiment is from 0g to 2010g with an increment of 20g (see in Figure-37). The sampling time is 30 seconds, which creates about 8~9 data points for each weight. More detailed data is shown in appendix.

The range covered in the first experiment is only 1/100 of the full range of the load cell with smaller increments and more data points for each weight. This experiment is designed to see how the data points float. The result shows no pattern but the data is between ±0.5g, which means the load cell has the resolution of 1g (equals to 0.005inch). This resolution is better than manually reading the CoCoRaHS rain gauge. Therefore, it can be assumed that the data is stable enough.

The second experiment covers a wide range, which is the maximum capacity of CoCoRaHS rain gauge. Its goal is to check the linearity of this 0-5kg load cell. The linearity will determine whether a correction to the database will be required. Through the result, it is concluded that this system provides enough linearity and, therefore, has no need for further correction.

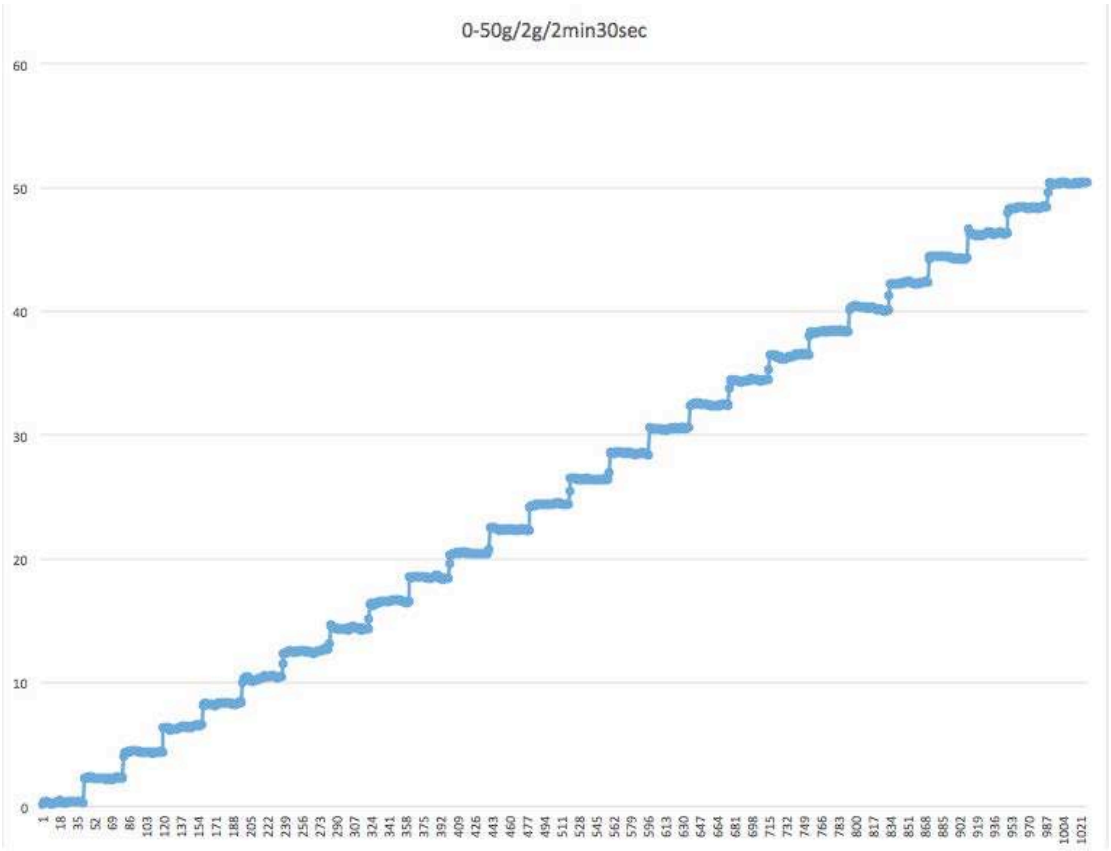



Figure-36 0g to 50g with 2g as increment

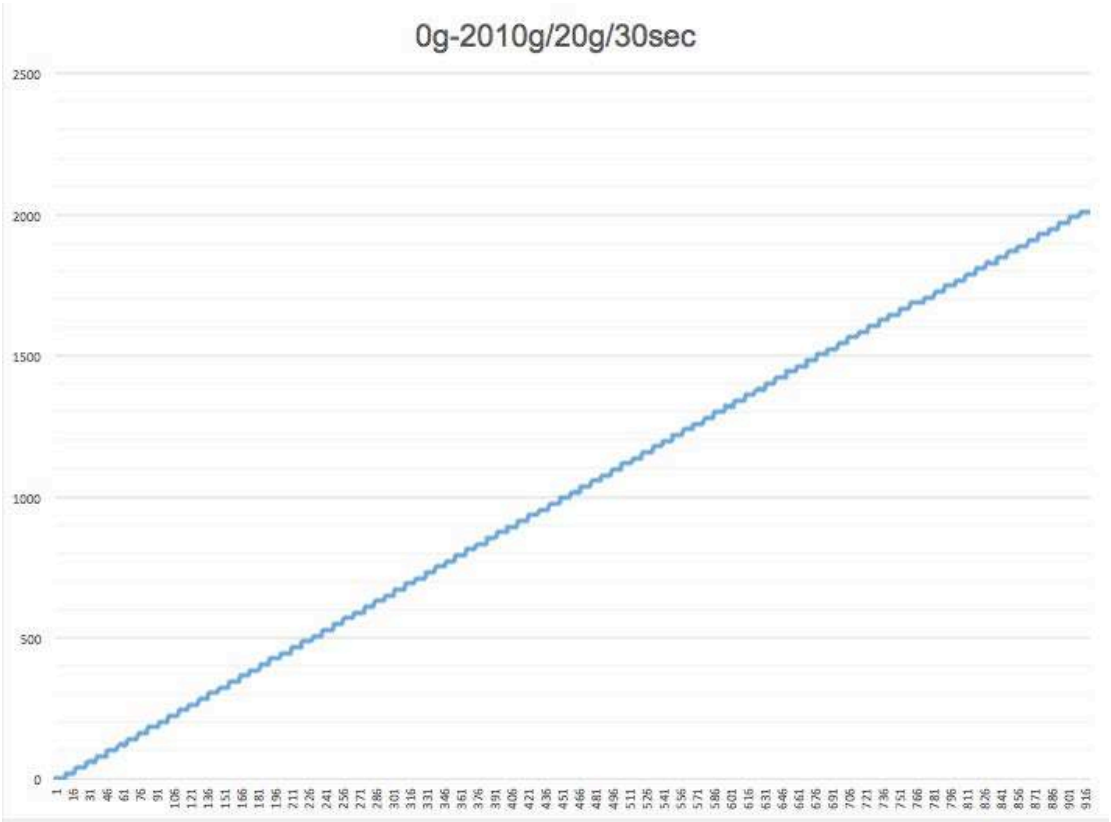



Figure-37 0g to 2010g with 20g as increment

# CHAPTER 5

# CONCLUSHION AND FUTURE

## 5.1 Proof-of-concept

This thesis has demonstrated the feasibility of an automated CoCoRaHS rain gauge. The tested weight-based automated rain gauge has enough resolution (about 0.5g equivalent to 0.0025inchs) and is capable of wirelessly transmitting precipitation data. The proof-of-concept rain gauge prototype could only be used in a laboratory environment and was not suitable for field testing.

The rain gauge tested in this thesis consists of the original CoCoRaHS cylinder, the modified CoCoRaHS bracket, and the new designed base with purchased electronic components. The purchased electronic components included two load cells, a HX711 ADC module, an Arduino, and an EMW3162 Wi-Fi module. One of the load cells came from a kitchen scale. Both load cells had their own bases in experiments. The 0-2kg load cell was mounted in a 3D-printed base. The shape of this 3D-printed base will be close to the real product design for the new CoCoRaHS rain gauge. A metal base built by the CSU machine shop was used for the 0-5kg load cell. This 0-5kg load cell satisfies the range requirement of the new rain gauge. The experiments were conducted in the CSU C105 Electrical Lab as well as other in-door locations.

There were five steps with some experiments designed to demonstrate the overall concept. In the first step, a kitchen scale with a 0-2kg load cell was reverse engineered. The 0-2kg load cell, the scale platform, and its PC board were used to demonstrate the weight-

based concept. The linearity between the weight input and electrical output demonstrated the possibility of the CoCoRaHS rain gauge being transforming into a weight-based rain gauge through a low-cost load cell. The second step was to design a new base for the present CoCoRaHS rain gauge. The base was designed and 3D printed. The third step was to replace the circuit from the kitchen scale with a design for the CoCoRaHS rain gauge. This circuit was designed to convert the weight of precipitation into the height of precipitation and transmit the data to other electronic devices wirelessly. An ADC module was selected as the bridge between the load cell and the microcontroller (an Arduino). The Wi-Fi shield made it easy to add a Wi-Fi module to the Arduino. The forth step was to program the Arduino and locate suitable receiver software for other electronic devices. After debugging, the code enabled the Arduino to obtain the digital data from the load cell. With 'Netassist' downloaded, computers, laptops and tablets are able to receive the data from the load cell wirelessly. The final step was to test the entire system. The entire system is defined as the 0-5kg load cell with the microelectronic circuit. The result from the system test showed this new weight-based automate rain gauge to be highly accurate and robust. Through the bill of materials and the estimated power calculations, it is believed that this new automated CoCoRaHS is sufficiently low-cost for amateurs to purchase and the battery power provides a useful lifetime of at least one-year.

## 5.2 Turn the design into a product

The next step after this proof-of-concept will be to turn this design into a product which could be used in field tests. For the reason that this design consists of both mechanical system

and electrical system, each will be separately discussed. In addition, the considerations to move forward with commercialization of the rain gauge will be described.

For mechanical system, the main concern will be the connection between the bracket and the base. There are two options to improve the stability. One is to manufacture the bracket and base as one piece. The problem will be the increasing cost to make this irregular, three-dimensional plastic part. Also the stress concentration near the joint should be carefully calculated which can influence the lifetime of the rain gauge. The other option is to find a best connection type, such as a screw connection for example. The screw is a strong connection but it depends on how many screws and where the screws are located. Over time, the user, especially in windy areas, may have to check the screws and may need to tighten the screws occasionally. In addition, the friction between the bracket and the overflow tube needs further study especially under freezing conditions. It is quite possible that the tube may freeze to the bracket, which will make the rain gauge produce incorrect measurements.

For the electrical system, the main development should be the coding and software. In this thesis the developed software consisted only to detect the weight change from the load cell and wirelessly transmit this data to a receiving laptop or tablet. The code did not include checking for unexpected weight change, like a bird landing on the cylinder. Another limitation is that the present code needs to be manually calibrated. The next step should make the whole process of data reading automatic with no manual intervention required. For software, the GUI (Graphical User Interface) should be developed for a better user

experience. The present GUI uses the software, Netassist, which is very basic and not convenient to use.

For the whole product, it will be very important that whole system is weatherproof. Freezing precipitation and wind will be the main factors that influence the data reading. The mechanical system should be wind-resistant while the electrical system needs to be waterproof.

## 5.3 Further possibilities

As discussed in the introduction of CoCoRaHS, the automated rain gauge could be used in more areas. The current design is based on the backyard situation. In order to cover more rural areas such as farms, islands, mountainous and deserts, the wireless method and power supply will need to be different from the present automated rain gauge. Wi-Fi is suitable for medium cover range, perhaps up to 100 meters. It may not bother the volunteer to change the back-yard battery every year, but this is probably not suitable for remote locations that are further away from a receiving station. Therefore, it might be more convenient to use cellular radio or satellite radio for long-range transmitting and self-powering systems for power, like a solar panel. A self-draining mechanical system should be designed for the reason that most remote areas are serviced only occasionally. In other cases, if the backyard volunteer does not want to change batteries, the rain gauge could also be designed with an electrical plug-in port. There are several options for power and communications to satisfy different situations.

REFERENCES


[1] "About Us." *CoCoRaHS-Community Collaborative Rain, Hail & Snow Network*. Colorado Climate Center. Web. 15 Mar. 2016. http://www.cocorahs.org/Media/docs/CoCoRaHSBrochure15FINAL.pdf

[2] Henry W. Reges*, Robert C. Cifelli, Nolan J. Doesken, and Julian Turner. "THE COMMUNITY COLLABORATIVE RAIN, HAIL AND SNOW NETWORK (COCORAHS): VOLUNTEERS MONITORING PRECIPITATION ACROSS THE NATION—THE NEXT STEP." *CoCoRaHS-Community Collaborative Rain, Hail & Snow Network*. Colorado Climate Center. Web. 15 Mar. 2016.

[3] "Fort Collins' 500 Year Flood (1997)." *Alabama WX Weather Blog*. The Alabama Weather Blog. Web. 20 Mar. 2016 http://www.alabamawx.com/?p=2725

[4] "Fort Collins: July 28, 1997." *The Weather and Climate Impact Assessment Science Program.* 2007 UCAR. Web. 15 Mar. 2016. http://www.assessment.ucar.edu/flood/flood_summaries/07_28_1997.html

[5] "CoCoRaHS-TrainingEPZ-English-Notes." *National Weather Service: Southern Region Headquarters.* National Weather Service. Web. 15 Mar. 2016. http://www.srh.noaa.gov/images/epz/mesonet/CoCoRaHS-TrainingEPZ-English-Notes.pdf

[6] "CoCoRaHS_Brochure." *CoCoRaHS-Community Collaborative Rain, Hail & Snow Network*. Colorado Climate Center. Web. 15 Mar. 2016. http://www.cocorahs.org/Media/docs/UT/CoCoRaHS_Brochure.pdf

[7] ROBERT CIFELLI, NOLAN DOESKEN, PATRICK KENNEDY, LAWRENCE D. CAREY, STEVEN A. RUTLEDGE, CHAD GIMMESTAD, AND TRACY DEPUE. "The Community Collaborative Rain, Hail, and Snow Network: Informal Education for Scientists and Citizens." *AMERICAN METEOROLOGICAL SOCIETY.* 2015 American Meteorological Society. Web. 15 Mar. 2016. http://journals.ametsoc.org/doi/pdf/10.1175/BAMS-86-8-1069

[8] CoCoRaHS-TrainingEPZ-English-Notes http://www.srh.noaa.gov/images/epz/mesonet/CoCoRaHS-TrainingEPZ-English-Notes.pdf

[9] Nolan J. Doesken. "TEN-YEAR COMPARISON OF DAILY PRECIPITATION FROM THE 4 INCH DIAMETER CLEAR PLASTIC RAIN GAUGE VERSUS THE 8 INCH DIAMETER METAL STANDARD RAIN GAUGE." *CoCoRaHS-Community Collaborative Rain, Hail & Snow Network*. Colorado Climate Center. Web. 15 Mar. 2016.

[10] "Datasheet: 3133 - Micro Load Cell (0-5kg) - CZL635." *RobotShop*. RobotShop inc. Web. 15 Mar. 2016. http://www.robotshop.com/media/files/pdf/datasheet-3133.pdf

[11] "Weight Sensor (Load Cell) 5KG." *Future Electronics.* Future Electronics Egypt (Arduino Egypt). Web. 15 Mar. 2016. http://store.fut-electronics.com/products/weight-sensor-load-cell-5kg

[12] "24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales." *Sparkfun.* SparkFun Electronics. Web. 15 Mar. 2016 https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

[13] "Overview." *ARDUINO & Genuion.* Arduino. Web. 15 Mar. 2016 https://www.arduino.cc/en/Main/ArduinoBoardUno

[14] "Datasheet: EMW3162." *MXCHIP.* MXCHIP Ltd. Web. 15 Mar. 2016 http://en.mxchip.com/uploadfiles/soft/20150309181614810 30.pdf

[15] "Weight Sensor for Arduino." *Layad Circuits.* Webzyme Softwares Pvt. Ltd. Web. 15 Mar. 2016 http://www.layadcircuits.com/tutorial.php?p=4

[16] "Feeding power to Arduino: the ultimate guide." *OPENELECTRONICS.* Open Electronics. Web. 15 Mar. 2016 http://www.open-electronics.org/the-power-of-arduino-this-unknown/

# APPENDIX A - DATASHEET

## HX711 Datasheet

海芯科技 AVIA Semiconductor

**HX711**

### 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales

#### DESCRIPTION

Based on Avia Semiconductor's patented technology, HX711 is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.

The input multiplexer selects either Channel A or B differential input to the low-noise programmable gain amplifier (PGA). Channel A can be programmed with a gain of 128 or 64, corresponding to a full-scale differential input voltage of ±20mV or ±40mV respectively, when a 5V supply is connected to AVDD analog power supply pin. Channel B has a fixed gain of 32. On-chip power supply regulator eliminates the need for an external supply regulator to provide analog power for the ADC and the sensor. Clock input is flexible. It can be from an external clock source, a crystal, or the on-chip oscillator that does not require any external component. On-chip power-on-reset circuitry simplifies digital interface initialization.

There is no programming needed for the internal registers. All controls to the HX711 are through the pins.

#### FEATURES

- **Two selectable differential input channels**
- **On-chip active low noise PGA with selectable gain of 32, 64 and 128**
- **On-chip power supply regulator for load-cell and ADC analog power supply**
- **On-chip oscillator requiring no external component with optional external crystal**
- **On-chip power-on-reset**
- **Simple digital control and serial interface: pin-driven controls, no programming needed**
- **Selectable 10SPS or 80SPS output data rate**
- **Simultaneous 50 and 60Hz supply rejection**
- **Current consumption including on-chip analog power supply regulator: normal operation < 1.5mA, power down < 1uA**
- **Operation supply voltage range: 2.6 ~ 5.5V**
- **Operation temperature range: -40 ~ +85℃**
- **16 pin SOP-16 package**

#### APPLICATIONS

- **Weigh Scales**
- **Industrial Process Control**

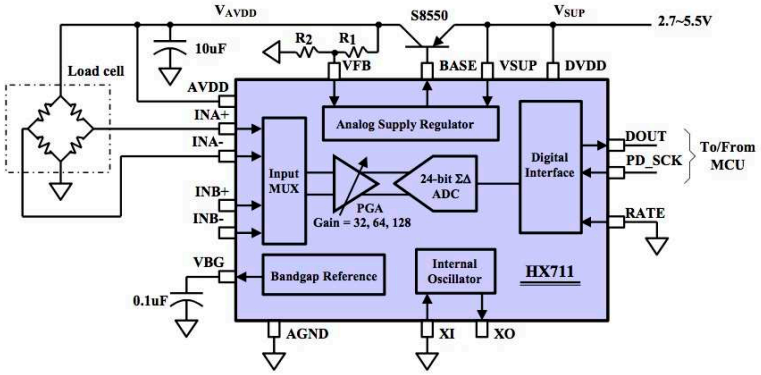
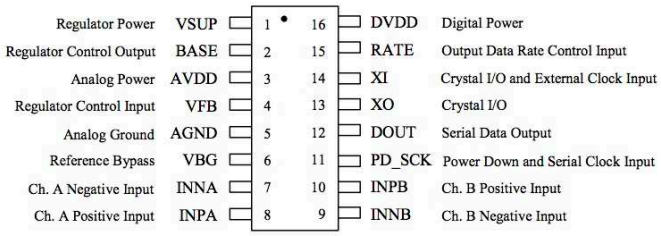



**Fig. 1 Typical weigh scale application block diagram**


TEL: (592) 252-9530 (P. R. China)
EMAIL: market@aviaic.com

## Pin Description




SOP-16L Package

| Pin # | Name | Function | Description |
|---|---|---|---|
| 1 | VSUP | Power | Regulator supply: 2.7 ~ 5.5V |
| 2 | BASE | Analog Output | Regulator control output（NC when not used） |
| 3 | AVDD | Power | Analog supply: 2.6 ~ 5.5V |
| 4 | VFB | Analog Input | Regulator control input（connect to AGND when not used） |
| 5 | AGND | Ground | Analog Ground |
| 6 | VBG | Analog Output | Reference bypass output |
| 7 | INA- | Analog Input | Channel A negative input |
| 8 | INA+ | Analog Input | Channel A positive input |
| 9 | INB- | Analog Input | Channel B negative input |
| 10 | INB+ | Analog Input | Channel B positive input |
| 11 | PD_SCK | Digital Input | Power down control (high active) and serial clock input |
| 12 | DOUT | Digital Output | Serial data output |
| 13 | XO | Digital I/O | Crystal I/O (NC when not used） |
| 14 | XI | Digital Input | Crystal I/O or external clock input, 0: use on-chip oscillator |
| 15 | RATE | Digital Input | Output data rate control, 0: 10Hz; 1: 80Hz |
| 16 | DVDD | Power | Digital supply: 2.6 ~ 5.5V |

**Table 1 Pin Description**

## KEY ELECTRICAL CHARACTERISTICS

| Parameter | Notes | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|
| Full scale differential input range | V(inp)-V(inn) | | ±0.5(AVDD/GAIN) | | V |
| Common mode input | | AGND+1.2 | | AVDD-1.3 | V |
| Output data rate | Internal Oscillator, RATE = 0 | | 10 | | Hz |
| | Internal Oscillator, RATE = DVDD | | 80 | | |
| | Crystal or external clock, RATE = 0 | | $f_{clk}$/1,105,920 | | |
| | Crystal or external clock, RATE = DVDD | | $f_{clk}$/138,240 | | |
| Output data coding | 2's complement | 800000 | | 7FFFFF | HEX |
| Output settling time(1) | RATE = 0 | | 400 | | ms |
| | RATE = DVDD | | 50 | | |
| Input offset drift | Gain = 128 | | 0.2 | | mV |
| | Gain = 64 | | 0.4 | | |
| Input noise | Gain = 128， RATE = 0 | | 50 | | nV(rms) |
| | Gain = 128， RATE = DVDD | | 90 | | |
| Temperature drift | Input offset（Gain = 128） | | ±6 | | nV/℃ |
| | Gain（Gain = 128） | | ±5 | | ppm/℃ |
| Input common mode rejection | Gain = 128， RATE = 0 | | 100 | | dB |
| Power supply rejection | Gain = 128， RATE = 0 | | 100 | | dB |
| Reference bypass（$V_{BG}$） | | | 1.25 | | V |
| Crystal or external clock frequency | | 1 | 11.0592 | 20 | MHz |
| Power supply voltage | DVDD | 2.6 | | 5.5 | V |
| | AVDD，VSUP | 2.6 | | 5.5 | |
| Analog supply current (including regulator) | Normal | | 1400 | | μA |
| | Power down | | 0.3 | | |
| Digital supply current | Normal | | 100 | | μA |
| | Power down | | 0.2 | | |

（1）Settling time refers to the time from power up, reset, input channel change and gain change to valid stable output data.

**Table 2 Key Electrical Characteristics**

### Analog Inputs

Channel A differential input is designed to interface directly with a bridge sensor's differential output. It can be programmed with a gain of 128 or 64. The large gains are needed to accommodate the small output signal from the sensor. When 5V supply is used at the AVDD pin, these gains correspond to a full-scale differential input voltage of ±20mV or ±40mV respectively.

Channel B differential input has a fixed gain of 32. The full-scale input voltage range is ±80mV, when 5V supply is used at the AVDD pin.

### Power Supply Options

Digital power supply (DVDD) should be the same power supply as the MCU power supply.

When using internal analog supply regulator, the dropout voltage of the regulator depends on the external transistor used. The output voltage is equal to $V_{AVDD}=V_{BG}*(R1+R2)/ R1$ (Fig. 1). This voltage should be designed with a minimum of 100mV below VSUP voltage.

If the on-chip analog supply regulator is not used, the VSUP pin should be connected to either AVDD or DVDD, depending on which voltage is higher. Pin VFB should be connected to Ground and pin BASE becomes NC. The external 0.1uF bypass capacitor shown on Fig. 1 at the VBG output pin is then not needed.

### Clock Source Options

By connecting pin XI to Ground, the on-chip oscillator is activated. The nominal output data rate when using the internal oscillator is 10 (RATE=0) or 80SPS (RATE=1).

If accurate output data rate is needed, crystal or external reference clock can be used. A crystal can be directly connected across XI and XO pins. An external clock can be connected to XI pin, through a 20pF ac coupled capacitor. This external clock is not required to be a square wave. It can come directly from the crystal output pin of the MCU chip, with amplitude as low as 150 mV.

When using a crystal or an external clock, the internal oscillator is automatically powered down.

### Output Data Rate and Format

When using the on-chip oscillator, output data rate is typically 10 (RATE=0) or 80SPS (RATE=1).

When using external clock or crystal, output data rate is directly proportional to the clock or crystal frequency. Using 11.0592MHz clock or crystal results in an accurate 10 (RTE=0) or 80SPS (RATE=1) output data rate.

The output 24 bits of data is in 2's complement format. When input differential signal goes out of the 24 bit range, the output data will be saturated at 800000h (MIN) or 7FFFFFh (MAX), until the input signal comes back to the input range.

### Serial Interface

Pin PD_SCK and DOUT are used for data retrieval, input selection, gain selection and power down controls.

When output data is not ready for retrieval, digital output pin DOUT is high. Serial clock input PD_SCK should be low. When DOUT goes to low, it indicates data is ready for retrieval. By applying 25~27 positive clock pulses at the PD_SCK pin, data is shifted out from the DOUT output pin. Each PD_SCK pulse shifts out one bit, starting with the MSB bit first, until all 24 bits are shifted out. The 25$^{th}$ pulse at PD_SCK input will pull DOUT pin back to high (Fig.2).

Input and gain selection is controlled by the number of the input PD_SCK pulses (Table 3). PD_SCK clock pulses should not be less than 25 or more than 27 within one conversion period, to avoid causing serial communication error.

| PD_SCK Pulses | Input channel | Gain |
|---|---|---|
| 25 | A | 128 |
| 26 | B | 32 |
| 27 | A | 64 |

**Table 3 Input Channel and Gain Selection**

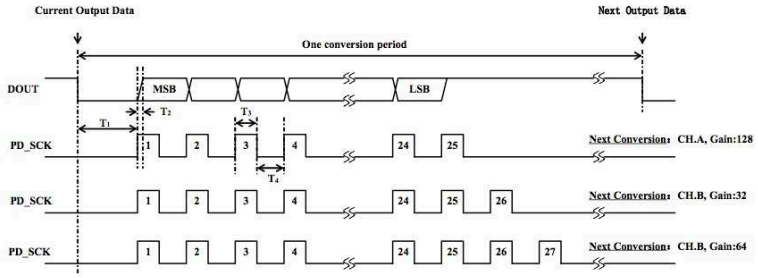



**Fig.2 Data output, input and gain selection timing and control**

| Symbol | Note | MIN | TYP | MAX | Unit |
|---|---|---|---|---|---|
| $T_1$ | DOUT falling edge to PD_SCK rising edge | 0.1 | | | μs |
| $T_2$ | PD_SCK rising edge to DOUT data ready | | | 0.1 | μs |
| $T_3$ | PD_SCK high time | 0.2 | 1 | 50 | μs |
| $T_4$ | PD_SCK low time | 0.2 | 1 | | μs |

## Reset and Power-Down

When chip is powered up, on-chip power on rest circuitry will reset the chip.

Pin PD_SCK input is used to power down the HX711. When PD_SCK Input is low, chip is in normal working mode.

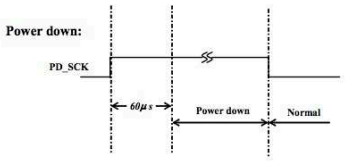



Fig.3 Power down control

When PD_SCK pin changes from low to high and stays at high for longer than 60μs, HX711 enters power down mode (Fig.3). When internal regulator is used for HX711 and the external transducer, both HX711 and the transducer will be powered down. When PD_SCK returns to low, chip will reset and enter normal operation mode.

After a reset or power-down event, input selection is default to Channel A with a gain of 128.

## Application Example

Fig.1 is a typical weigh scale application using HX711. It uses on-chip oscillator (XI=0), 10Hz output data rate (RATE=0). A Single power supply （2.7～5.5V） comes directly from MCU power supply. Channel B can be used for battery level detection. The related circuitry is not shown on Fig. 1.

**Reference PCB Board (Single Layer)**

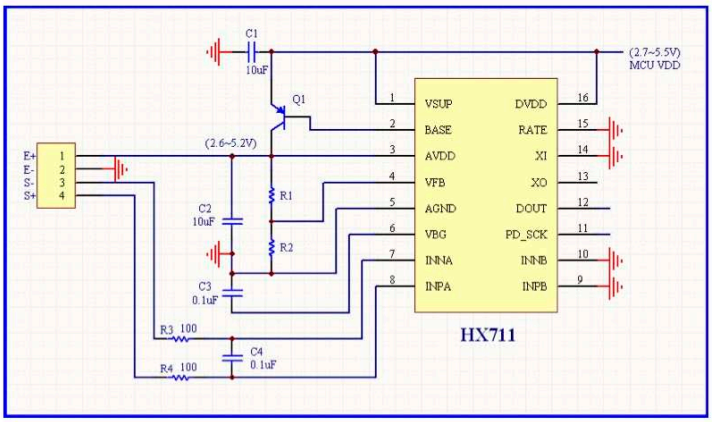



**Fig.4 Reference PCB board schematic**

**Fig.5 Reference PCB board layout**

**Reference Driver (Assembly)**

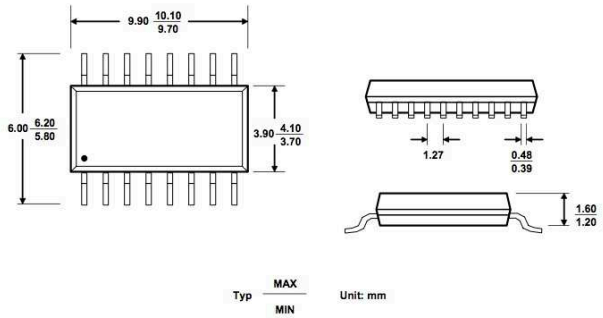
```
/*----------------------------------------------------------------
Call from ASM:      LCALL   ReaAD
Call from C:    extern unsigned long ReadAD(void);
                   .
                   .
                  unsigned long data;
                  data=ReadAD();
                   .
                   .
----------------------------------------------------------------*/
PUBLIC        ReadAD
HX711ROM      segment  code
rseg          HX711ROM

sbit           ADDO = P1.5;
sbit           ADSK = P0.0;
/*-----------------------------------------------
 OUT:   R4, R5, R6, R7   R7=>LSB
-----------------------------------------------*/
ReadAD:
    CLR    ADSK              //AD Enable (PD_SCK set low)
    SETB   ADDO              //Enable 51CPU I/O
    JB     ADDO,$            //AD conversion completed?
    MOV    R4,#24
ShiftOut:
    SETB   ADSK              //PD_SCK set high (positive pulse)
    NOP
    CLR    ADSK              //PD_SCK set low
    MOV    C,ADDO            //read on bit
    XCH    A,R7              //move data
    RLC    A
    XCH    A,R7
    XCH    A,R6
    RLC    A
    XCH    A,R6
    XCH    A,R5
    RLC    A
    XCH    A,R5
    DJNZ   R4,ShiftOut        //moved 24BIT?
    SETB   ADSK
    NOP
    CLR    ADSK
    RET
    END
```

**Reference Driver（C）**

```
//-----------------------------------------------------------------
sbit   ADDO = P1^5;
sbit   ADSK = P0^0;
unsigned long ReadCount(void){
  unsigned long Count;
  unsigned char i;
  ADDO=1;
  ADSK=0;
  Count=0;
  while(ADDO);
  for (i=0;i<24;i++){
    ADSK=1;
    Count=Count<<1;
    ADSK=0;
    if(ADDO) Count++;
  }
  ADSK=1;
  Count=Count^0x800000;
  ADSK=0;
  return(Count);
}
```

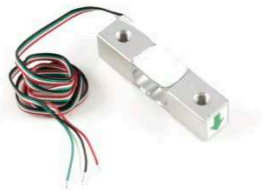
**Package Dimensions**




SOP-16L Package

# Datasheet

## 3133 - Micro Load Cell (0-5kg) - CZL635



**Contents**



## What do you have to know?

A load cell is a force sensing module - a carefully designed metal structure, with small elements called strain gauges mounted in precise locations on the structure. Load cells are designed to measure a specific force, and ignore other forces being applied. The electrical signal output by the load cell is very small and requires specialized amplification. Fortunately, **the 1046 PhidgetBridge will perform all the amplification and measurement of the electrical output.**

Load cells are designed to measure force in one direction. They will often measure force in other directions, but the sensor sensitivity will be different, since parts of the load cell operating under compression are now in tension, and vice versa.

## How does it work - For curious people

Strain-gauge load cells convert the load acting on them into electrical signals. The measuring is done with very small resistor patterns called strain gauges - effectively small, flexible circuit boards. The gauges are bonded onto a beam or structural member that deforms when weight is applied, in turn deforming the strain-gauge. As the strain gauge is deformed, it's electrical resistance changes in proportion to the load.

The changes to the circuit caused by force is much smaller than the changes caused by variation in temperature. Higher quality load cells cancel out the effects of temperature using two techniques. By matching the expansion rate of the strain gauge to the expansion rate of the metal it's mounted on, undue strain on the gauges can be avoided as the load cell warms up and cools down. The most important method of temperature compensation involves using multiple strain gauges, which all respond to the change in temperature with the same change in resistance. Some load cell designs use gauges which are never subjected to any force, but only serve to counterbalance the temperature effects on the gauges that measuring force. Most designs use 4 strain gauges, some in compression, some under tension, which maximizes the sensitivity of the load cell, and automatically cancels the effect of temperature.

## Installation

This Single Point Load Cell is used in small jewelry scales and kitchen scales. It's mounted by bolting down the end of the load cell where the wires are attached, and applying force on the other end **in the direction of the arrow.** Where the force is applied is not critical, as this load cell measures a shearing effect on the beam, not the bending of the beam. If you mount a small platform on the load cell, as would be done in a small scale, this load cell provides accurate readings regardless of the position of the load on the platform.

## Calibration

A simple formula is usually used to convert the measured mv/V output from the load cell to the measured force:

$$Measured\ Force = A * Measured\ mV/V + B \text{ (offset)}$$

It's important to decide what unit your measured force is - grams, kilograms, pounds, etc.

This load cell has a rated output of 1.0±0.15mv/v which corresponds to the sensor's capacity of 5kg.

To find A we use

$$Capacity = A * Rated\ Output$$

$$A = Capacity / Rated\ Output$$

$$A = 5 / 1.0$$

$$A = 5$$

Since the Offset is quite variable between individual load cells, it's necessary to calculate the offset for each sensor. Measure the output of the load cell with no force on it and note the mv/V output measured by the PhidgetBridge.

$$Offset = 0 - 5 * Measured\ Output$$

## Product Specifications

| | |
|---|---|
| **Mechanical** | |
| Housing Material | Aluminum Alloy |
| Load Cell Type | Strain Gauge |
| Capacity | 5kg |
| Dimensions | 55.25x12.7x12.7mm |
| Mounting Holes | M5 (Screw Size) |
| Cable Length | 550mm |
| Cable Size | 30 AWG (0.2mm) |
| Cable - no. of leads | 4 |
| **Electrical** | |
| Precision | 0.05% |
| Rated Output | 1.0±0.15 mv/V |
| Non-Linearity | 0.05% FS |
| Hysteresis | 0.05% FS |
| Non-Repeatability | 0.05% FS |
| Creep (per 30 minutes) | 0.1% FS |
| Temperature Effect on Zero (per 10°C) | 0.05% FS |
| Temperature Effect on Span (per 10°C) | 0.05% FS |
| Zero Balance | ±1.5% FS |
| Input Impedance | 1130±10 Ohm |
| Output Impedance | 1000±10 Ohm |
| Insulation Resistance (Under 50VDC) | ≥5000 MOhm |
| Excitation Voltage | 5 VDC |
| Compensated Temperature Range | -10 to ~+40°C |
| Operating Temperature Range | -20 to ~+55°C |
| Safe Overload | 120% Capacity |
| Ultimate Overload | 150% Capacity |

## Glossary

**Capacity**
The maximum load the load cell is designed to measure within its specifications.

**Creep**
The change in sensor output occurring over 30 minutes, while under load at or near capacity and with all environmental conditions and other variables remaining constant.

**FULL SCALE or FS**
Used to qualify error - FULL SCALE is the change in output when the sensor is fully loaded. If a particular error (for example, Non-Linearity) is expressed as 0.1% F.S., and the output is 1.0mV/V, the maximum non-linearity that will be seen over the operating range of the sensor will be 0.001 mV/V. An important distinction is that this error doesn't have to only occur at the maximum load. If you are operating the sensor at a maximum of 10% of capacity, for this example, the non-linearity would still be 0.001mV/V, or 1% of the operating range that you are actually using.

**Hysteresis**
If a force equal to 50% of capacity is applied to a load cell which has been at no load, a given output will be measured. The same load cell is at full capacity, and some of the force is removed, resulting in the load cell operating at 50% capacity. The difference in output between the two test scenarios is called hysteresis.

**Excitation Voltage**
Specifies the voltage that can be applied to the power/ground terminals on the load cell. In practice, if you are using the load cell with the PhidgetBridge, you don't have to worry about this spec.

**Input Impedance**
Determines the power that will be consumed by the load cell. The lower this number is, the more current will be required, and the more heating will occur when the load cell is powered. In very noisy environments, a lower input impedance will reduce the effect of Electromagnetic interference on long wires between the load cell and PhidgetBridge.

**Insulation Resistance**
The electrical resistance measured between the metal structure of the load cell, and the wiring. The practical result of this is the metal structure of the load cells should not be energized with a voltage, particularly higher voltages, as it can arc into the PhidgetBridge. Commonly the load cell and the metal framework it is part of will be grounded to earth or to your system ground.

**Maximum Overload**
The maximum load which can be applied without producing a structural failure.

**Non-Linearity**
Ideally, the output of the sensor will be perfectly linear, and a simple 2-point calibration will exactly describe the behaviour of the sensor at other loads. In practice, the sensor is not perfect, and Non-linearity describes the maximum deviation from the linear curve. Theoretically, if a more complex calibration is used, some of the non-linearity can be calibrated out, but this will require a very high accuracy calibration with multiple points.

**Non-Repeatability**
The maximum difference the sensor will report when exactly the same weight is applied, at the same temperature, over multiple test runs.

**Operating Temperature**
The extremes of ambient temperature within which the load cell will operate without permanent adverse change to any of its performance characteristics.

**Output Impedance**
Roughly corresponds to the input impedance. If the Output Impedance is very high, measuring the bridge will distort the results. The PhidgetBridge carefully buffers the signals coming from the load cell, so in practice this is not a concern.

**Rated Output**
Is the difference in the output of the sensor between when it is fully loaded to its rated capacity, and when it's unloaded. Effectively, it's how sensitive the sensor is, and corresponds to the gain calculated when calibrating the sensor. More expensive sensors have an exact rated output based on an individual calibration done at the factory.

**Safe Overload**
The maximum axial load which can be applied without producing a permanent shift in performance characteristics beyond those specified.

**Compensated Temperature**
The range of temperature over which the load cell is compensated to maintain output and zero balance within specified limits.

**Temperature Effect on Span**
Span is also called rated output. This value is the change in output due to a change in ambient temperature. It is measured over 10 degree C temperature interval.

**Temperature Effect on Zero**
The change in zero balance due to a change in ambient temperature. This value is measured over 10 degree C temperature interval.

**Zero Balance**
Zero Balance defines the maximum difference between the +/- output wires when no load is applied. Realistically, each sensor will be individually calibrated, at least for the output when no load is applied. Zero Balance is more of a concern if the load cell is being interfaced to an amplification circuit - the PhidgetBridge can easily handle enormous differences between +/-. If the difference is very large, the PhidgetBridge will not be able to use the higher Gain settings.

## APPENDIX B – ARDUINO LIBRARY

Resource: https://codebender.cc/library/HX711#README.md

### HX711.cpp

```
#include <Arduino.h>
#include <HX711.h>

HX711::HX711(byte dout, byte pd_sck, byte gain) {
	PD_SCK	= pd_sck;
	DOUT	= dout;

	pinMode(PD_SCK, OUTPUT);
	pinMode(DOUT, INPUT);

	set_gain(gain);
}

HX711::~HX711() {

}

bool HX711::is_ready() {
	return digitalRead(DOUT) == LOW;
}

void HX711::set_gain(byte gain) {
	switch (gain) {
		case 128:		// channel A, gain factor 128
			GAIN = 1;
			break;
		case 64:		// channel A, gain factor 64
			GAIN = 3;
			break;
		case 32:		// channel B, gain factor 32
			GAIN = 2;
			break;
	}

	digitalWrite(PD_SCK, LOW);
	read();
```

```
}
long HX711::read() {
	// wait for the chip to become ready
	while (!is_ready());

	byte data[3];

	// pulse the clock pin 24 times to read the data
	for (byte j = 3; j--;) {
		for (char i = 8; i--;) {
			digitalWrite(PD_SCK, HIGH);
			bitWrite(data[j], i, digitalRead(DOUT));
			digitalWrite(PD_SCK, LOW);
		}
	}

	// set the channel and the gain factor for the next reading using the clock pin
	for (int i = 0; i < GAIN; i++) {
		digitalWrite(PD_SCK, HIGH);
		digitalWrite(PD_SCK, LOW);
	}

	data[2] ^= 0x80;

	return ((uint32_t) data[2] << 16) | ((uint32_t) data[1] << 8) | (uint32_t) data[0];
}

long HX711::read_average(byte times) {
	long sum = 0;
	for (byte i = 0; i < times; i++) {
		sum += read();
	}
	return sum / times;
}

double HX711::get_value(byte times) {
	return read_average(times) - OFFSET;
}

float HX711::get_units(byte times) {
	return get_value(times) / SCALE;
}
```

```
void HX711::tare(byte times) {
	double sum = read_average(times);
	set_offset(sum);
}

void HX711::set_scale(float scale) {
	SCALE = scale;
}

void HX711::set_offset(long offset) {
	OFFSET = offset;
}

void HX711::power_down() {
	digitalWrite(PD_SCK, LOW);
	digitalWrite(PD_SCK, HIGH);
}

void HX711::power_up() {
	digitalWrite(PD_SCK, LOW);
}
```

HX711.h

```
#ifndef HX711_h
#define HX711_h

#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

class HX711
{
	private:
		byte PD_SCK;	// Power Down and Serial Clock Input Pin
		byte DOUT;	// Serial Data Output Pin
		byte GAIN;	// amplification factor
		long OFFSET;	// used for tare weight
```

```
        float SCALE;      // used to return weight in grams, kg, ounces, whatever

    public:
        // define clock and data pin, channel, and gain factor
        // channel selection is made by passing the appropriate gain: 128 or 64 for channel A, 32 for channel B
        // gain: 128 or 64 for channel A; channel B works with 32 gain factor only
        HX711(byte dout, byte pd_sck, byte gain = 128);

        virtual ~HX711();

        // check if HX711 is ready
        // from the datasheet: When output data is not ready for retrieval, digital output pin DOUT is high. Serial clock
        // input PD_SCK should be low. When DOUT goes to low, it indicates data is ready for retrieval.
        bool is_ready();

        // set the gain factor; takes effect only after a call to read()
        // channel A can be set for a 128 or 64 gain; channel B has a fixed 32 gain
        // depending on the parameter, the channel is also set to either A or B
        void set_gain(byte gain = 128);

        // waits for the chip to be ready and returns a reading
        long read();

        // returns an average reading; times = how many times to read
        long read_average(byte times = 10);

        // returns (read_average() - OFFSET), that is the current value without the tare weight; times = how many readings to do
        double get_value(byte times = 1);

        // returns get_value() divided by SCALE, that is the raw value divided by a value obtained via calibration
        // times = how many readings to do
        float get_units(byte times = 1);

        // set the OFFSET value for tare weight; times = how many times to read the tare value
        void tare(byte times = 10);

        // set the SCALE value; this value is used to convert the raw data to "human readable"
```

```
data (measure units)
        void set_scale(float scale = 1.f);
        // set OFFSET, the value that's subtracted from the actual reading (tare weight)
        void set_offset(long offset = 0);

        // puts the chip into power down mode
        void power_down();

        // wakes up the chip after power down mode
        void power_up();
};

#endif /* HX711_h */
```

README.md

HX711
=====

An Arduino library to interface the Avia Semiconductor HX711 24-Bit Analog-to-Digital Converter (ADC) for Weight Scales.

This is my humble attempt at creating an Arduino library for this ADC:
http://www.dfrobot.com/image/data/SEN0160/hx711_english.pdf

Other libraries exist, including this very good one, which I first used and which is the starting point for my library:
https://github.com/aguegu/ardulibs/tree/master/hx711

Although other libraries exist, I needed a slightly different approach, so here's how my library is different than others:

1. It provides a tare() function, which "resets" the scale to 0. Many other implementations calculate the tare weight when the ADC is initialized only. I needed a way to be able to set the tare weight at any time. Use case: place an empty container on the scale, call tare() to reset the readings to 0, fill the container and get the weight of the content.

2. It provides a power_down() function, to put the ADC into a low power mode. According to the datasheet, "When PD_SCK pin changes from low to high and stays at high for longer than 60µs, HX711 enters power down mode". Use case: battery powered scales. Accordingly, there is a power_up() function to get the chip out of the low power mode.

3. It has a set_gain(byte gain) function that allows you to set the gain factor and select the channel. According to the datasheet, "Channel A can be programmed with a gain of 128 or 64, corresponding to a full-scale differential input voltage of ±20mV or ±40mV respectively, when a 5V supply is connected to AVDD analog power supply pin. Channel B has a fixed gain of 32.". The same function is used to select the channel A or channel B, by passing 128 or 64 for channel A, or 32 for channel B as the parameter. The default value is 128, which means "channel A with a gain factor of 128", so one can simply call set_gain(). Also, the function is called from the constructor.

4. The constructor has an extra parameter "gain" that allows you to set the gain factor and channel. The constructor calls the "set_gain" function mentioned above.

5. The "get_value" and "get_units" functions can receive an extra parameter "times", and they will return the average of multiple readings instead of a single reading.

How to Calibrate Your Scale

1. Call set_scale() with no parameter.
2. Call tare() with no parameter.
3. Place a known weight on the scale and call get_units(10).
4. Divide the result in step 3 to your known weight. You should get about the parameter you need to pass to set_scale.
5. Adjust the parameter in step 4 until you get an accurate reading.