LIVESTOCK TEMPERATURE MONITOR

By

Cain Benink

Michael Goldstein

Yue Wang

Final Report for ECE 445, Senior Design, Spring 2017

TA: Daniel Gardner

3 May 2017

Project No. 4

Abstract

In the United States alone, millions of livestock die each year from preventable illness.[1] Current means for diagnosing sick animals are largely manual focusing on visible signs of illness like runny noses, clouded eyes, and loss of appetite.[2] We aim to greatly reduce livestock loss by creating an inexpensive device which will constantly monitor livestock temperatures and alert farmers of animals with fevers, which not only are more accurate indicators of illness, but also occur days before other symptoms. Our device resides within a normal livestock ear tag and features electronics to measure temperature. It periodically broadcasts the temperature data over the 915MHz Industrial, Scientific, and Medical (ISM) band to a receiver which will provide a user friendly interface for the herd's temperatures and which animals are running fevers. We initially intend to target domestic beef cattle as they are the animal for which loss is most costly to farmers. Our device will save farmers money by reducing animal loss, providing farmers with data they need for granular application of antibiotics, keeping more animals certified organic, and reducing veterinary costs. The long-term implication is that the cost to raise livestock will decrease lowering meat prices worldwide.

Contents

Prefix A Table of Acronyms and Abbreviations

Acronym / Abbreviation	Meaning
ADC	Analog to Digital Converter
AWS	Amazon Web Services
C	Celsius
CPU	Central Processing Unit
cc	Cubic Centimeter
CRC	Cyclic Redundancy Check
DB	Database
DC	Direct Current
g	Gram
GUI	Graphical User Interface
ID	Identifier
ISM	Industrial, Scientific, and Medical

kΩ	Kilo Ohms
MHz	Megahertz
mm	Millimeter
NA	Network Analyzer
Ω	Ohms
PCB	Printed Circuit Board
RAM	Random Access Memory
RF	Radio Frequency
RTC	Real-time Clock
Rx	Receive / Receiver
TI	Texas Instruments
TTL	Transistor-Transistor Logic
Tx	Transmit / Transmitter

1 Introduction

1.1 Objective and Background

Current methods of diagnosing sick animals are insufficient. The process is largely manual. We plan to change that by offering a solution to wirelessly monitor the core temperature of cattle.

According to the Wall Street Journal, “To monitor cattle health, feedlots typically rely on cowboys to ride through pens, watching for lethargic animals that are having trouble breathing.”[2] Because the process is labor intensive and the symptoms identifiable show up days after the onset of disease, the livestock industry suffers from a great deal of both false positive and false negative sickness. This results in a great deal of animal death, over medication of animals, unnecessary vet visits, and unnecessary loss of organic certification. This problem costs the cattle industry over 2 billion dollars in losses annually. As of 2016, there are about 92.0 million heads of cattle in the United States.[4] Of these cows, digestive and respiratory illness, mastitis, and other diseases kill approximately 1.5 million of them each year.[4]

We aim to fix these problems by creating a cow tag that will constantly measure the temperature of the animal and send it back to a receiver via RF 915MHz, known as the Industrial, Scientific, and Medical (ISM) band. We chose this band because it is unregulated in the United States and offers the best transmission range and bandwidth of the unregulated RF bands. The receiver would upload data via the internet and a graphical user interface (GUI) would alert the farmer of animals that are running fevers. This solution leads to less labor intensive, more accurate, and earlier diagnosis of sick animals. Because of the amount of money farmers lose to this problem, a working solution would be easily marketable -- buying our product would literally save them money.

1.2 High Level Requirements

- Units must be durable enough and have a battery life long enough for the unit to last the entire lifespan of a beef cow (minimum 18 months).
- Units must transmit temperatures successfully every hour with a resolution of at least 0.2 degrees Celsius in order to detect a cow’s fever within 12 hours.
- Units must have a bulk cost of less than $10.

1.3 Block Diagram

1.3.1 Tag Block Diagram

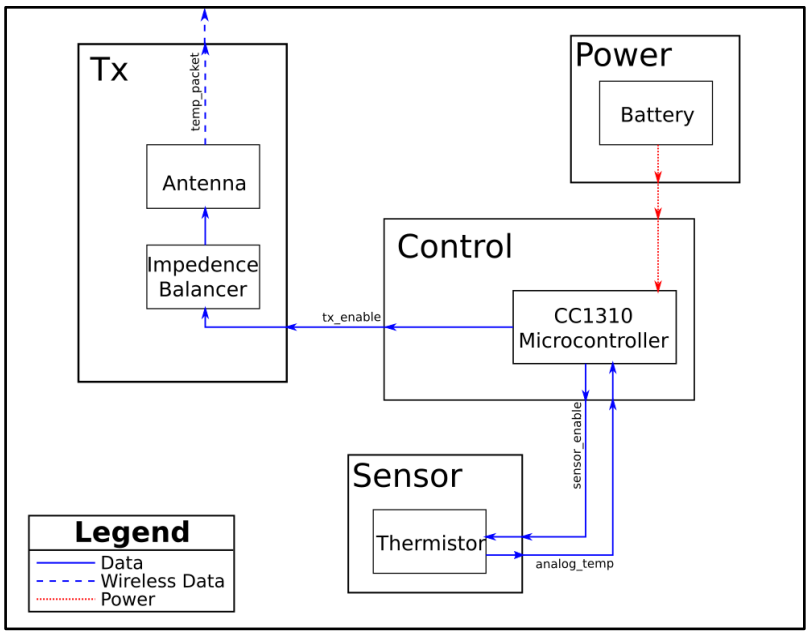

Figure 1.1: Block Diagram for Tag

As illustrated in Figure 1.1, the Tag unit has four main modules: the power module, the control module, the sensor module, and the transmission module. The main component of the power module is the battery. It is used to provide power to the control module which in turn provides control signals and power to the rest of the blocks. The control block is composed of a Texas Instruments (TI) CC1310 microcontroller. The control signals are enabled or disabled by software. The sensor module is a thermistor in a simple voltage divider circuit with a resistor. It is used to generate an analog signal for the temperature around the probe. The transmission unit is composed of an antenna and an impedance balancing network to convert the imaginary impedance output of the control unit into the impedance expected by the antenna.

1.3.2 Receiver Block Diagram

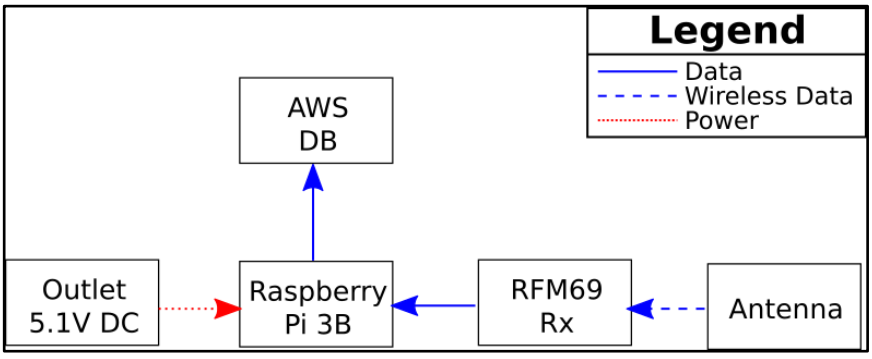

Figure 1.2: Block Diagram of Receiver

Our receiver unit is composed of five modules. As seen in Figure 1.2, RF packets are absorbed in the antenna and carried to the receiver. We chose the RFM69 transceiver unit, but to save development costs, ended up using the Texas Instruments CC1310 “Launchpad” Development Board instead since we already had the part for testing the tag unit. The receiver communicates with a Raspberry Pi 3B over serial interface. The Raspberry Pi in turn uploads the data to an Amazon Web Services (AWS) Database.

1.3.3 Physical Specifications

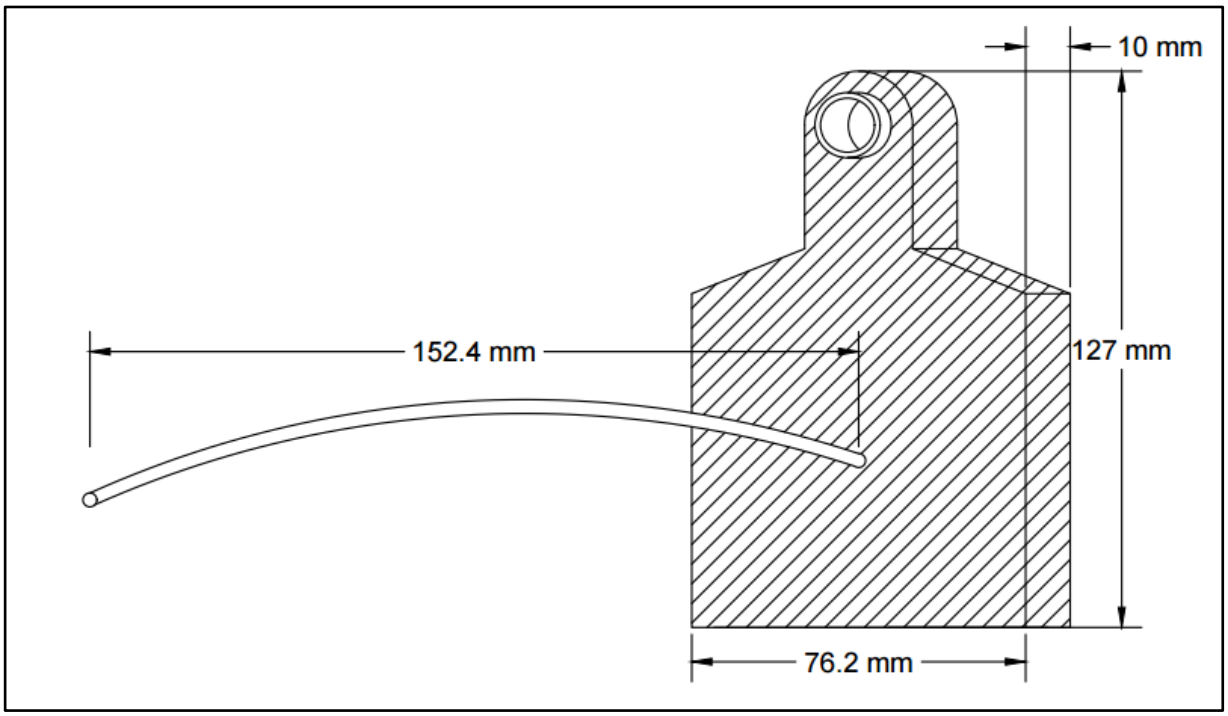

Figure 1.3: Physical CAD Drawing of Cow Tag

Figure 1.3 illustrates the physical dimensions of a cow tag thermometer. The tag must be a total of 5 inches (127mm) tall, 3 inches (76.2mm) wide, and 10mm deep. The thermistor protrudes out of the plastic housing 6 inches in order to reach the ear canal of the cow. The 10 mm depth is to have enough room for approximately 2.5mm of electronics plus enough cushioning to protect the electronics. The hole at the top of the tag is for the piercing that the cow receives. The piercer keeps the tag connected to the cow’s ear.

2 Design

2.1 Design Procedure

2.1.1 Sensor Block

We knew that the basic sensor we were going to use would be a resistor in series with a thermistor so that the voltage divider output would be calculable later. We considered using a thermocouple for the temperature sensor for the added resolution and lower power usage, but we decided to stick with a thermistor because the power usage met the requirement, it was cheaper, it was more linear in our temperature range and it was easier to get readings from.

We found an equation online for the resistance of a thermistor based on temperature, we will use this equation later to determine which exact thermistor and resistor we will use in series for the sensor unit.

$$R = R_o e^{-B/T_o} e^{B/T} \quad (1)$$

2.1.2 Control Block

Our first idea for controlling the circuit logic was to use transistor-transistor logic (TTL) logic gates because we were familiar with them, but this did not turn out to be the best option for our design due to high complexity and power usage that did not meet the specification. The option for our design that we chose was to use a low power microcontroller. After evaluating many microcontrollers, we chose the Texas Instruments CC1310. We chose this microcontroller because of its ability to transmit at 915Mhz, low standby power usage, package size and low cost.

2.1.3 Power Block

In our design, the power module must provide 3V of power, have a large enough capacity to last 18 months, and have a small enough form factor to fit in a cow ear tag. We decided to use a coin cell battery, the CR2025, because it meets both requirements. We did some conservative power calculations that gave us the minimum power we needed from the battery and chose a battery with an order of magnitude more amp hours than the minimum. We could have used solar power to power our device, but we decided that it was too expensive for our high-level requirement and the ear tags are usually too dirty and disposable for solar power usage.

2.1.4 Transmitter Block

The transmission block is responsible for preparing the message for sending and then sending the message. For our design the CC1310 chip reads the voltage data from the voltage division on the sensor and calculates a temperature from the reading. It then packages the 2-byte temperature with a 4-byte preamble (used to synchronize the transmitter and receiver), 2-byte magic number, 2-byte unique ID and 2-byte cyclic redundancy check (CRC).

We then broadcast that message on 915Mhz via our trace antenna. Preceding the trace antenna lie our components to match the impedance of the antenna to the impedance of the chip. This is necessary because it will keep the messages from rebounding back and forth between the antenna and the chip and this will minimize noise and maximize power.

2.1.5 Receiver Block

The receiver unit is responsible for collecting data from all the various ear tag transmitters and processing them to be present to the customer. It includes an antenna attached to a RF Receiver chip to receive signals, a Raspberry Pi 3 to process and store the data, and a surge protector connected to a conventional outlet for power.

2.2 Design Details

2.2.1 Sensor Block

We picked a thermistor at random to test its tolerance. The properties of the thermistor is shown below.

Table 2.1: Important Specifications of the Thermistor [5]

25 C Resistance	47 kΩ
25 C B Value	4050 K
Tolerance	1%
Operating Temperature Range	-40 C to +125 C

Using these numbers and equation (1) we can calculate the resistance range of the thermistor over 38.6 to 40 C, the average temperatures of a healthy and sick cow respectively.

$R_{min} = 47000\Omega \; x \; e^{-4050K/298.15K} e^{4050K/313K} = 24672\Omega$

(2)

$R_{max} = 47000\Omega \; x \; e^{-4050K/298.15K} e^{4050K/311.6K} = 26149\,\Omega$

(3)

We averaged these two values to determine that the resistance of the resistor should be 25.5 kΩ for the 47 kΩ thermistor. With these two resistors determines, we can now model the behavior of the voltage divider output with respect to temperature.

Our voltage divider circuit is shown below:

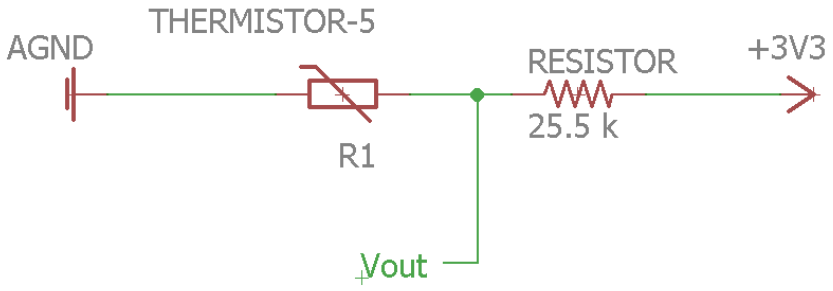

Figure 2.1: Sensor Block Schematic

And rearranging equation (1) and using the voltage divider formula we get:

$$V_{out} = \frac{R}{R+25500} \quad (4)$$

$$T = \frac{B}{ln(R/R_o) - B/T_o} \quad (5)$$

Using these two formulas we can then model the output of the voltage divider as shown in Figure 2.2 below. We can see that for the temperature range of 38.6 to 40 C, the voltage divider output will be around 1.475 and 1.52 V.

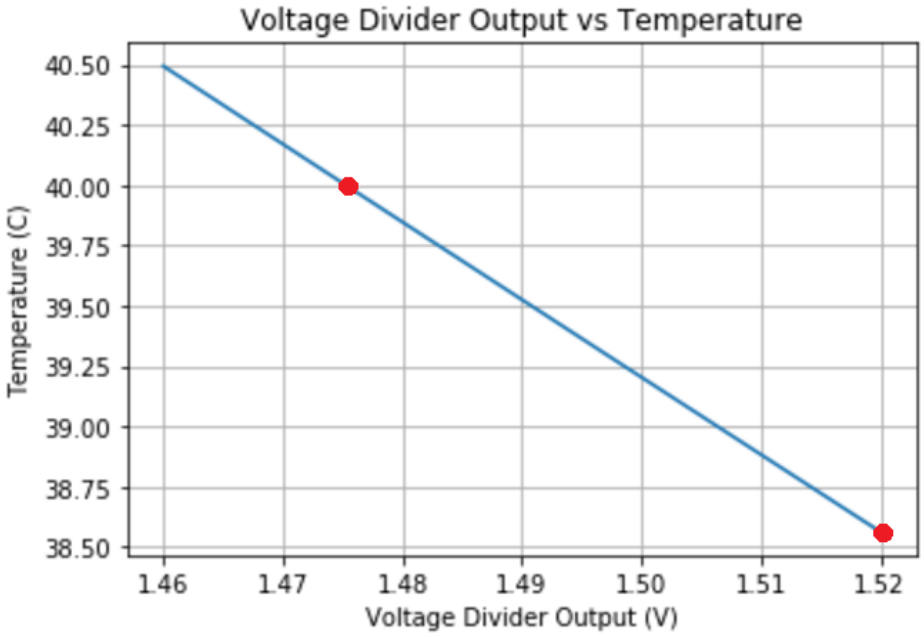

Figure 2.2: Voltage Output vs Thermistor Temperature

2.2.2 Control Block

The control logic keeps track of time so that it knows when to power up the other components, and also controls how long the power and data is being sent to the other components. We also have a 2 byte identifier value stored in the microchip upon programming for signal identification as well as another 2 byte identifier value for livestock identification. That value along with a 4-byte preamble, 2 byte temperature and a 2 byte CRC is transmitted to the receiver (so the entire packet is a total of 12 bytes). The control block itself is built into a TI CC1310[6] chip, which is a microcontroller that features a built-in ISM band RF transceiver and 12-bit analog to digital converter (ADC). The control block gets 3V of power from the battery.

The RTC is also built into the CC1310 chip and allows scheduling of configurable periodic interrupts that are used to schedule reading from the sensor block and transmitting the data. The RTC is kept to a 32.8 kHz clock cycle which means that it needs to count to 32800 * 60 seconds / minute * 5 = 9840000 in order to keep track of the 5 minute intervals between data transmission.

The analog to digital converter is also built into the CC1310 and receives the voltage output from the sensor module and reads it into the microcontroller. The CC1310 ADC is 12 bits, and taking into consideration the Vcc of 3V, leaves an error range of +/- 366uV.

2.2.3 Power Block

The calculations for how much power is needed to power the transmitter unit for two years is as follows:

Important Microcontroller Specifications[6]:

- TX at +10 dBm 868 MHz: 13.4 mA
- Standby: 0.7 µA (RTC Running and RAM and CPU Retention)

Message is 12 bytes.
Send message every (s) seconds (s) = sec/msg
Standby in between messages
Wake up time is 174 us
Conservatively we are assuming a 1 ms wake up and set up time.

Standby power:
Conservatively it's constantly using standby power.

$$.7\mu A * 2\ years\ * 365\ days\ *\ 24\ hours\ =\ P_{standby}\ =\ .0123Ah$$

(6)

Message Transfer Power Usage:

$$\frac{12\ bytes}{50kb/s} =\ .002\ s/msg$$

$$.002\ s/msg\ *\ 13.4mA\ =\ 26.8\ \mu A/msg$$

$$\frac{1}{(s)}\ *\ 26.8\ \mu A/msg\ =\ \frac{26.8}{(s)}\mu A$$

$$\frac{26.8}{(s)}\ \mu A/msg\ *\ 2\ years\ *\ 365\ days\ *\ 24\ hours\ =\ P_{tx}$$= .470/(s) Ah (7)

Wake up/Set up Time Power Consumption:

$$1\ ms/msg\ *\ \frac{1}{(s)}\ msg/s\ =\ Duty\ Cycle\ =\ \frac{.001}{(s)}$$

$$\frac{.001}{(s)}\ *\ 2.5mA\ =\ \frac{2.5}{(s)}\ \mu A$$

$$\frac{2.5}{(s)}\ \mu A\ *\ 2\ years\ *\ 365\ days\ *\ 24\ hours\ =\ P_{Wake}\ =\ .0438/(s)\ Ah$$

(8)

Active Power Usage:

$$\frac{1}{(s)} msg/s\ *\ .00192\ s/msg\ =\ Duty\ Cycle\ =\ .00192/(s)$$

$$\frac{.00192}{(s)}\ *\ 2.5mA\ =\ \frac{4.8}{(s)}\ \mu A$$

$$\frac{4.8}{(s)}\mu A\ *\ 2\ years\ *\ 365\ days\ *\ 24\ hours\ =\ P_{Active}\ =\ .0841/(s)\ Ah$$

(9)

Thermistor Power Usage During Transfer:

$$25.5\ k\Omega\ (Thermistor)\ +\ 25.5\ k\Omega\ (Series\ Resistance)\ =\ Resistance$$

$$3.3V\ =\ Voltage$$

$$\frac{3.3\ V}{51\ k\Omega}\ =\ 64.7\ \mu A$$

$$.00192/(s)\ =\ Duty\ Cycle$$

$$\frac{.00192}{(s)}\ *\ 64.7\ \mu A\ =\ 124.2/(s)\ nA$$

$$\frac{124.2}{(s)} nA\ *\ 2\ years\ *\ 365\ days\ *\ 24\ hours\ = P_{thermistor} =\ .002176/(s)\ Ah$$

(10)

Total Amp-hours:

Total Ah = (6) + (10) + (7) + (8) + (9)

$$.0123\ +\ \frac{.6001}{(s)}\ Ah\ =\ Total\ Amp-hours \qquad (11)$$

Since we are planning on having the device transmit every 5 minutes, then (s) would be 300, so total power consumption in two years using equation 11 is:

$$.0123\ +\ \frac{.6001}{300}\ Ah\ =\ 14.3\ mAh$$

(12)

Thus, the transmitter needs a battery that outputs 3V and has at least 14.30 mAh, and button batteries satisfy that condition. We chose the Panasonic CR2025 which has output of 3V and capacity of 165 mAh, it can be purchased for as cheap as 30 cents and widely available everywhere[8]. That particular button battery will allow the transmitter to theoretically run up to 20 years.

2.2.4 Transmitter Block

The RF transmitter is built into the CC1310 chip. Among the variety of available frequencies that the CC1310 chip is able to transmit at, we chose 915 MHz because it is unregulated. The RF transmitter takes the output of the ADC, prepends a 2-byte magic number and a 2-byte ID as a header, and appends a 2-byte CRC to form the full 8-byte packet.

Table 2.2 Packet Structure

Bytes 0-3	Bytes 4-5	Bytes 6-7	Bytes 8-9	Bytes 10-11
Preamble (0xAAAAAAAA)	Magic Number (0xBEEF)	Tag ID Number	Temperature (in form of ADC block output)	2 byte CRC

Since we picked the transmission to be 915 MHz, the antenna is a 915 MHz capable antenna. We used a meandering monopole trace antenna because among trace antennas it boasts the best efficiency with

the main tradeoff being area on the printed circuit board (PCB) [7]. Because our overall physical device is significantly larger than the PCB needs to be, we had sufficient room for a large trace antenna.

2.2.5 Receiver Block

In the initial receiver design, we intended to use the RFM69 transceiver unit to receive the 915MHz packets. In the end, we chose to use the TI CC1310 Launchpad Development Board instead to save on development cost because we already had the part. It performs functionally the same since the both transmit data the received data to a computer over a serial interface. To receive the data from the receiver, we intended to write python software on a Raspberry Pi. Because Python is cross-platform, we chose to use a laptop computer instead to save on development cost.

One of the more important functions of the Receiver Unit is to translate the ADC output back into a human-readable temperature. We were able to do that by using Equation 5 from Section 2.2. The final code can be seen below in Figure 2.3.

```
import serial
import datetime
import math

print('\n\nCowMon v0.1')
print('    by Michael Goldstein, Yue Wang, and Cain Benink\n')
print('Listening for incoming packets on ttyACM0...\n')

def toFTemp(t):
    if (v > 1):
        return -1;
    return 4050.0/math.log(-430326*v/(v-1)) - 273.15

ser = serial.Serial('/dev/ttyACM0', 9600, timeout=None)
while(True):
    packet = ser.read(6)
    magic = int.from_bytes(packet[0:2], byteorder='little')
    tagid = int.from_bytes(packet[2:4], byteorder='little')
    temp = int.from_bytes(packet[4:6], byteorder='little')
    ftemp = toFTemp(temp)
    print("Packet Received at %s" % str(datetime.datetime.now()).split('.')[0])
    print("    Magic:     %04X" % magic)
    print("    ID:        %04X" % tagid)
    print("    Temp:      %04X" % temp)
    print("    convTemp:  %f\n" % ftemp)
```

Figure 2.3 Receiver Python Code

During the design, we had planned to have the receiver upload the data to an AWS database. Due to time constraints, we were unable to setup a database and make the receiver upload to it. Instead, to have data persist on the filesystem, the output of the python script was piped into a file where that would be processed and displayed in a GUI.

2.3 Verification

2.3.1 Sensor Block

For the sensor block to meet our requirements, it has to output the voltage range shown in figure 2.2. At 38.6 deg C we want it to output 1.52V and at 40 deg C we want it to output 1.475V. The sensor must also be accurate by .2 deg C across that same temperature range. The sensor also has to output linearly

between these values. Lastly, the sensor must reach out 150mm from its base.

Verification of the length is easy because we bought a thermistor that was 150mm and measured it to be 150mm in the lab with a ruler. To test the correct output voltages we set up figure 2.1 on a breadboard and took voltage readings at different temperatures.

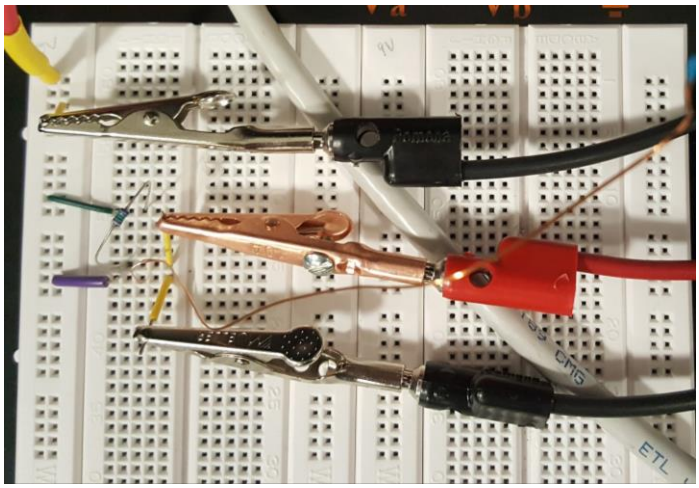

Figure 2.3: Sensor Unit Testing on Breadboard

Figure 2.4: Sensor Unit Measurements

Using the measurements in Figure 2.4, we calculated to see if the voltage reading of the thermistor matched the temperature reading of the infrared temperature sensor. Combining equation (4) and (5), we simplified them down so we could obtain temperature from voltage.

$$R = \frac{8500\ xV_{out}}{1+V_{out}/3} \tag{13}$$

$$T = \frac{B}{ln(R/R_o)+B/T_o} = \frac{4050}{ln(0.18085xV_{out})-ln(1-V_{out}/3)+4050/298.15} \tag{14}$$

Plugging our voltage reading into equation (14), we got the temperature to be 23.3 C, which is a lot different than the 24.5 C that the infrared temperature sensor measured. We then unplugged the power supply and measured each component to see if we have the right values. The resistor gave us the

expected 25.5 kΩ while the thermistor was at 49 kΩ, and plugging that value into equation (5) gave us a temperature of 24.24 C which is within our requirement range of 0.2 C away from 24.5 C. We then concluded that attaching a separate power supply on the breadboard affected the circuit, but we were unable to get the complete PCB working so we were unable to test the sensor unit with a coin cell battery instead.

2.3.2 Control Block

To verify that the control block was working we used the debugger on the dev board to verify it was receiving data from the thermistor, packaged it with the magic number, ID and CRC and sent it out for transmitting. To do this we ran our software with the debugger running and fed a voltage input into the GPIO that was for the thermistor. We made sure the output IO pin to the antenna was a 12-byte packet that contained the 4-byte preamble (used to synchronize the transmitter and receiver), 2-byte magic number “0xBEEF”, 2-byte temperature data, 2-byte ID and 2-byte CRC.

2.3.3 Power Block

To verify the power block was working we tested if the ground plane and power pins were getting the 3V3 and we tested power usage when transmitting and on standby. When we looked at the current usage when the the circuit was transmitting, the circuit was drawing 13.4mA which matched what we calculated in the power calculations. We were unable to get the chip to go into standby mode and thus were not able to fully verify the power consumption requirements.

2.3.4 Transmitter Block

For the transmitter to match our requirements, we need to make sure that the impedances of both the antenna itself and the network connecting the antenna to the microprocessor are 50 Ω. This is to ensure that there are no power loss due to unnecessary reactances and that the signal passing through will reflect backwards as little as possible.

First we tuned the antenna. We made sure that there were no other components attached to the antenna, and then we connected a SMA wire to the antenna pad. We then used a Network Analyzer (NA) to scan the antenna’s frequency response so that the S11 parameter which is a variable in determining the input reflection, is as low as possible.

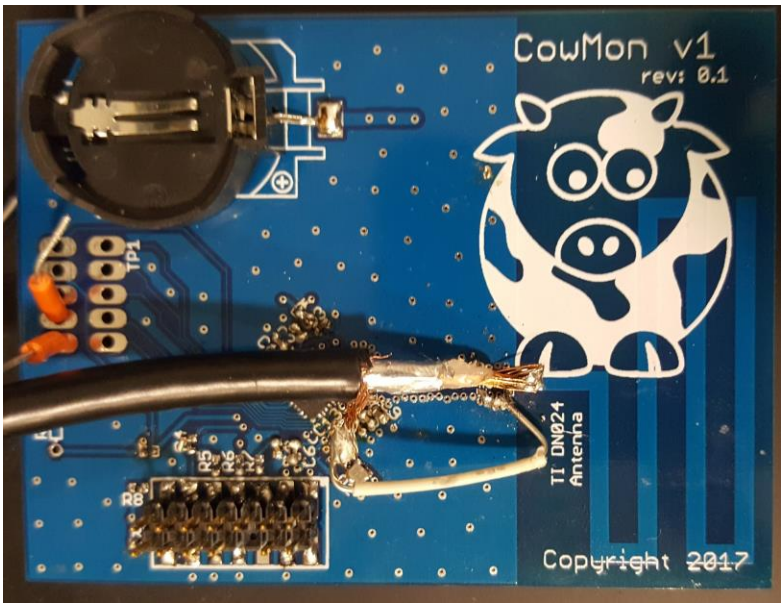

Figure 2.5: Antenna Tuning Connection

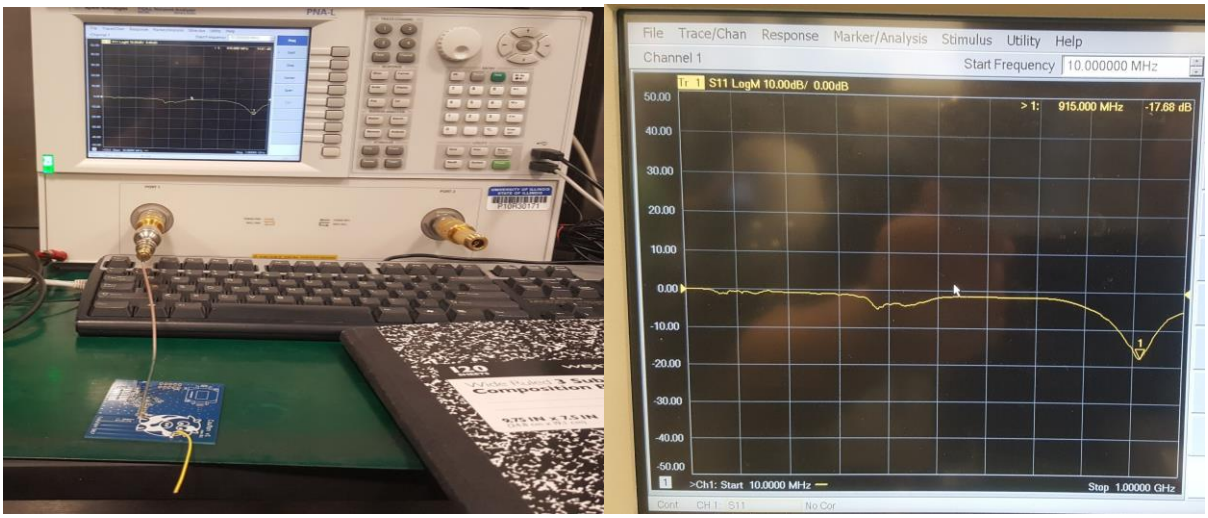

Figure 2.6: Antenna Tuning Results

Figure 2.6 above is the pictures of a successful match. As visible on the left, by adding another wire onto our existing antenna and trimming it down, we eventually got the dip in S11 value to 915 MHz, meaning that we have the lowest possible S11 value when transmitting at 915 MHz.

Then we had to do the component matching. Here we cut the trace connecting the antenna to the rest of the pcb, and attached a SMA wire going into the network connecting to the antenna.

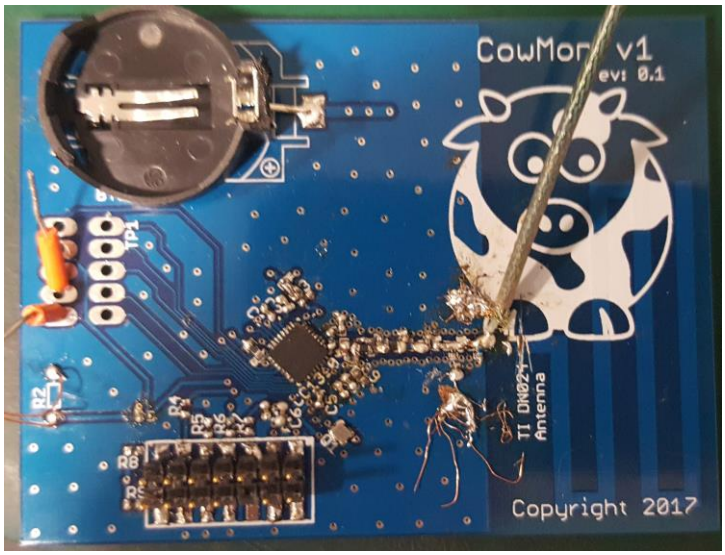

Figure 2.7: Component Matching Connection

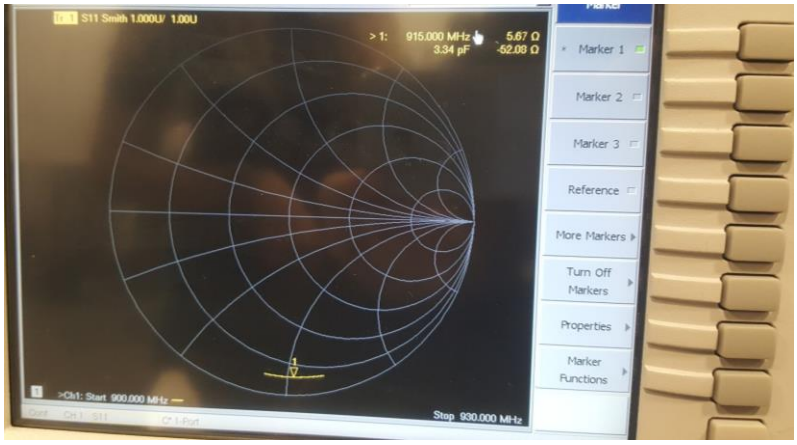

Figure 2.8: Initial Network Analyzer Measurements

We can then measure the network on the NA, with the results in Figure 2.8 above. We need to get the marker at 915 MHz to the center of the Smith chart, so we use Advance Design Systems software to simulate what would happen if we added a matching network to our existing network.

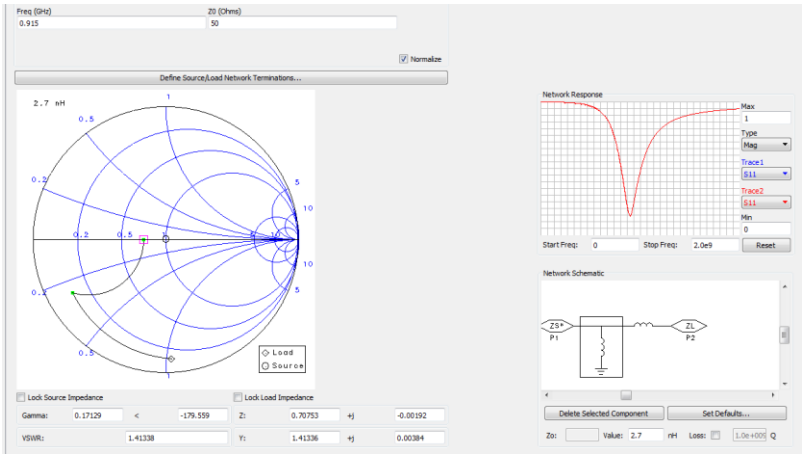

Figure 2.9: Achievable Component Impedance Correction

We see from Figure 2.9 that with the part values we currently possess, we can make a matching network using two inductors that will bring the network on the pcb pretty close to 50 Ω. We then run another simulation using the additional matching network to make sure that these results are correct.

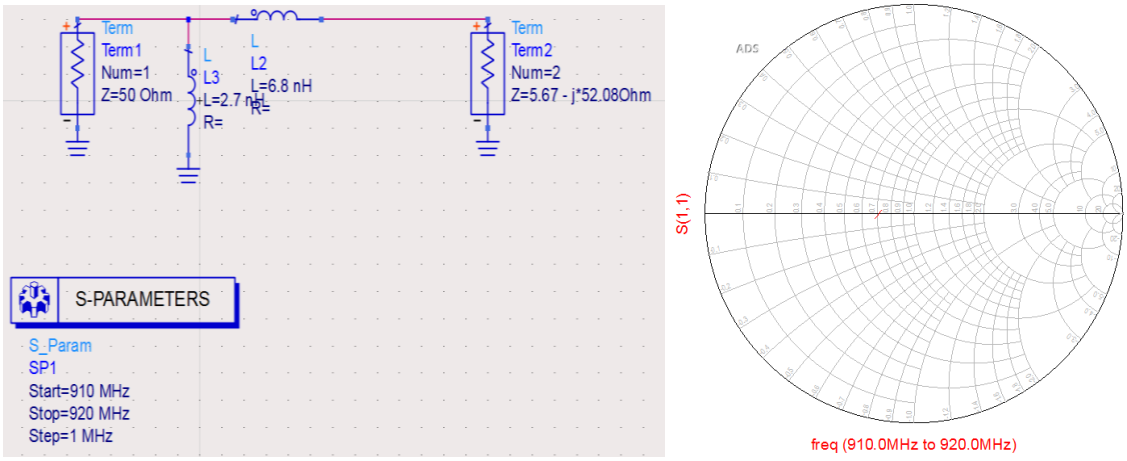

Figure 2.10: ADS Simulation of Matching Network Left and Simulation Results Right

On the left figure of Figure 2.10, the antenna is represented by the box on the left side with a resistance of 50 Ω, that was what we tuned it to earlier. On the right figure of Figure 2.10 is another box with the impedance of our network that we measured on the NA. We run the simulation for a signal between 910 and 920 MHz, and see that on the right, the small red dash is where we expected it to be.

We then installed this matching network onto our PCB, and remeasured it on the NA. Unfortunately, the results did not match our simulations.

Figure 2.11: After Matching Network Results

We can see from Figure 2.11 that the 915 MHz marker is not where we expected it to be, so we removed the matching network and tried again. Our second attempt was also met with failure, and at that point we had to stop testing because some of the pads on the PCB fell off and we could no longer solder. Our initial plan was to solder three full boards, but unfortunately the orders were not completed properly and we only ended up with some of the parts we needed. Our teaching assistant helped us find some of the parts we needed, but in the end we only had enough parts to finish soldering two complete boards. Since we needed one board untouched by testing to show the complete design, after the one we tested on had its pad fall off we were unable to continue component matching.

2.3.5 Receiver Block

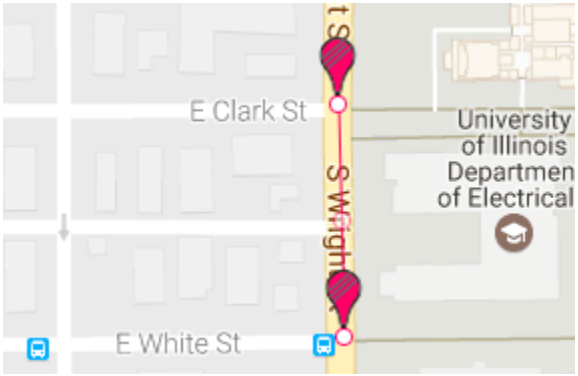

Figure 2.12: Measuring Distance Between Test Points

The receiver block requirements are found in Table A.5. Requirement 1 was verified by first developing software on a second TI CC1310 Launchpad Development Board to periodically transmit packets at 915MHz. As seen in Figure 2.12, using a Google Maps ruler, we measured out a distance between two points outside the ECE building that were about 106m apart.

Figure 2.13: Conducting the Receiver Verification

As seen in Figure 2.13, a group member then stood at each measured endpoint: one with the tag unit and one with the receiver unit. We then verified that the receiver unit was successfully receiving the packets.

The same test also verified Requirement 2. The receiver software outputs the contents of the packet which we were able to manually compare with what was transmitted. Requirement 3 was verified by feeding test known ADC readings into the Python software's conversion function and verifying that the output was correct.

3 Costs

3.1 Parts

Table 3.1 Parts Costs - Tag Unit

Manufacturer	Part Number	Part Type	Unit Px 100	Unit Px 5000	Supplier
Texas Instruments	CC1310F32RHBR	Microcontroller	$3.740	$2.200	TI Store
Panasonic	CR2025	Battery	$0.222	$0.159	Mouser
Linx	BAT-HLD-001	Battery Holder	$0.233	$0.183	Mouser
Murata	NXFT15WB473FA2B150	Thermistor	$0.274	$0.172	Mouser
Panasonic	ERJ-2RKF2552X	Resistor	$0.006	$0.003	Mouser
Murata	81-BLM18HE152SN1D	Ferrite Beads	$0.102	$0.602	Mouser
Murata	478-9839-1-ND	Capacitor	$0.750	$0.750	Digi-Key
Murata	490-1320-1-ND	Capacitor	$0.100	$0.100	Digi-Key
Murata	490-6328-1-ND	Capacitor	$0.100	$0.100	Digi-Key
Murata	490-5922-1-ND	Capacitor	$0.100	$0.100	Digi-Key
Murata	90-5948-1-ND	Capacitor	$0.100	$0.100	Digi-Key
Murata	490-5939-1-ND	Capacitor	$0.100	$0.100	Digi-Key
Murata	490-5872-1-ND	Capacitor	$0.100	$0.100	Digi-Key
Murata	490-5934-1-ND	Capacitor	$0.100	$0.100	Digi-Key
Murata	490-7282-1-ND	Capacitor	$0.100	$0.100	Digi-Key
Murata	445-1023-1-ND	Inductor	$0.190	$0.190	Digi-Key
Murata	490-2628-1-ND	Inductor	$0.100	$0.100	Digi-Key
Murata	490-6846-1-ND	Inductor	$0.180	$0.180	Digi-Key
Murata	490-1142-1-ND	Inductor	$0.170	$0.170	Digi-Key
Vishay Dale	541-25.5KLCT-ND	Resistor	$0.100	$0.100	Digi-Key
Vishay Dale	541-4.70KLCT-ND	Resistor	$0.100	$0.100	Digi-Key
Vishay Dale	311-100LRCT-ND	Resistor	$0.100	$0.100	Digi-Key
Vishay Dale	541-22JCT-ND	Resistor	$0.100	$0.100	Digi-Key
Total			**$7.167**	**$6.009**	

Table 3.2 Machining & Fabrication Costs - Tag Unit

Part Type	Quote Description	Unit Px 100	Unit Px 5000	Supplier
PCB	50x40mm, 1.6mm thick, 6mil spacing	$4.900	$0.931	PCBWay
Housing	Plastic Case, 3D printed ABS	$3.938*	$3.938*	Illinois MakerLab
Total		**$8.838**	**$4.869**	

*Estimate based on $0.15/g * 1.05g/cc * 25cc

Table 3.3 Unit & Lot Costs - Tag Unit

Unit Price, 100	$16.005
Unit Price, 5000	$10.878
Lot Price, 100	$1,600.500
Lot Price, 5000	$5,439.000

Table 3.4 Parts Costs - Receiver Unit

Manufacturer	Part Number	Part Type	Unit Px 1	Unit Px 100	Supplier
Raspberry Pi	Pi 3 Model B	Computer	$39.950	$39.950	Adafruit
Johanson	0915AT43A0026E	Antenna	$1.280	$0.574	Mouser
Kingston	SDC10/4GB	Micro SD Card	$6.490	$6.490	Amazon
CanaKit	Pi MicroUSB Supply	Power Supply	$7.990	$7.990	Amazon
Sparkfun	COM-13909	RF Transmitter	$4.950	$4.950	Mouser
Total			**$60.66**	**$59.954**	

Table 3.5 Unit & Lot Costs - Receiver Unit

Unit Price, 1	$60.660
Unit Price, 100	$59.954
Lot Price, 100	$5,995.400

3.2 Labor

3.2.1 Research and Development

We have three team members doing research and development for approximately 20 hours per week for about 14 weeks. 2.5 * 3*20*14*$45 yields a research and development cost of approximately $94,500.

3.2.2 Assembly

Assuming that each of us can assemble 15 tag units per hour, assembly adds $3 ($45/15) to the unit cost. This changes the 5,000 lot unit price to $10.586 per unit and the total lot price to $52,930.

Assuming that each of us can assemble 3 receiver units per hour, assembly adds $15 ($45/15) to the unit cost. This changes the 100 lot unit price to $74.954 per unit and the total lot price to $7,495.

3.3 Grand Total

The grand total cost to research, develop, order parts, and assemble a 5,000 lot of tag units and a 100 lot of receiver units is $94,500 + $52,930 + $7,495, which comes out to a total of $154,925.

4 Ethical Considerations

4.1 Relevant IEEE Ethics Guide Provisions[10]:

1) to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;

This ethic guideline is important to us because what we are working on is very tightly related to public health. Catching sick cows before they are used for meat or milk produce is an important part of our society.

3) to be honest and realistic in stating claims or estimates based on available data;

It is important for us to not make any false or inaccurate claims in the design, research and testing that we do for our project. These will only hurt us and other engineers around us.

6) to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;

This project is aimed at improving all of our technical competence and knowledge to better our understanding of our field.

7) to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;

Throughout our design process we will be asking for honest criticism of our work by peers, TAs and professors. We must be able to accept criticism of our work even if we do not agree. Looking at our project from multiple points of view will help in the growth of our technical and business skills.

10) to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

During the duration of the design process it is very important for us to assist and support each other in our work. We are here to improve our technical and general engineering skills and we can do that better by working together as a team.

5 Conclusion

Our project, Livestock Temperature Monitor, proved to be an excellent learning opportunity. Many of the difficulties we faced were not anticipated. None of us were familiar with RF theory and, in turn, developing the PCB and tuning the antenna proved to be a much greater challenge than expected. Soldering the PCBs was also a huge challenge and likely prevented us from being able to successfully program the microcontroller on the PCB.

Despite challenges, we accomplished a great deal. We were able to get an approximately matched circuit for the trace antenna on our PCB. We wrote embedded software that successfully read an analog temperature and transmitted data over RF 915MHz to a receiver. We wrote receiver software that could transform the voltage reading back into a temperature. We also gained a great amount of insight as to what future work needs to be done in order to make this project commercially viable.

Quite a bit of future work would be needed to make our product usable. We would need to successfully program the microcontroller on the PCB. Redoing the layout of the PCB to use larger parts where possible and adding more test pads for programming pins would help with this. Also, using better soldering irons with a microscope would better enable us to successfully solder small components. A tag would need to be designed in CAD and 3D printed so that the PCB could be inside. The trace antenna would need to be rematched assuming that the tag and a cow's ear are part of the antenna so that in the real world the antenna still has maximum gain at 915MHz. An AWS database would need to be setup to hold the data from the receiver. Lastly, software algorithms would need to run on the data in order to detect which animals are running fevers and a GUI would need to be created to clearly present this information to the end user.

Alternative designs could go with an approach other than an eartag. Perhaps an RFID-like chip could be implanted that would transmit temperature data to a receiver located inside a feeder any time a cow goes to eat. Running a scientific experiment of our completed project would be necessary to determine if the ear tag approach would be successful with an actual cow.

References

[1]"Cattle Death Loss", Usda.mannlib.cornell.edu, 2011. [Online]. Available: http://usda.mannlib.cornell.edu/usda/current/CattDeath/CattDeath-05-12-2011.pdf. [Accessed: 04- May- 2017].

[2] J. Bunge, "High-Tech Tagging Comes to the Ranch", WSJ, 2017. [Online]. Available: https://www.wsj.com/articles/high-tech-tagging-comes-to-the-ranch-1474630330. [Accessed: 09- Feb- 2017].

[3] "Beef Industry Statistics - Beef USA." Beef Industry Statistics - Beef USA. National Cattlemen's Beef Association, 2015. [Online]. [Accessed: 08- Feb- 2017].

[4] National Agricultural Statistics Service (NASS), Agricultural Statistics Board, United States Department of Agriculture (USDA). “Cattle Death Loss.” USDA. National Agricultural Statistics Service. 2011, May 12. [Online]. 11 Nov. 2016. [Accessed: 08- Feb- 2017].

[5] "Thermistor", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Thermistor#B_or_.CE.B2_parameter_equation. [Accessed: 22- Feb- 2017].

[6] CC1310 SimpleLink™ Ultra-Low-Power Sub-1 GHz Wireless MCU, 1st ed. Dallas, Texas: Texas Instruments, 2015. [Accessed: 22- Feb- 2017].

[7]T. Instruments, Antenna Selection Quick Guide, 1st ed. Dallas, Texas: Texas Instruments, 2013.

[8] "Panasonic CR2025 3V Lithium Coin Battery", *Brooklyn Battery Works*, 2017. [Online]. Available: http://www.brooklynbatteryworks.com/panasonic-cr2025-3v-lithium-coin-battery/?gclid=CNixv6WLqtIC. [Accessed: 25- Feb- 2017].

[9]T. Meteer, "WHAT IS COLD WEATHER TO A COW?", *University of Illinois Extension*, 2013. [Online]. Available: https://web.extension.illinois.edu/oardc/eb275/entry_7747/. [Accessed: 25- Feb- 2017].

[10] IEEE Corporate. “IEEE Code of Conduct.” IEEE Governance. Institute of Electrical and Electronics Engineers. http://www.IEEE.org, 2017. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 06- Feb- 2017]

Appendix A Requirements and Verifications

A.1 Sensor Module

Table A.1: Sensor Module Requirements and Verification

Requirement	Verification
1. The Sensor output a voltage that is related to the sensed temperature. a. The Sensor shall output a voltage range of .1V, +/- .01V, linearly for the temperature range 38.6 C to 40 C respectively. 2. The Sensor shall be able to reach into a cow's ear. a. The sensor shall be able to reach 150mm, +/- 10mm, from the ear tag.	1. The sensor shall be able to accurately measure temperature. a. The sensor will be put in a temperature chamber. The chamber temperature will be varied from 38.6 C to 40 C over 4 hours, starting at 38.6 and increasing at a rate of .35 C/hour. We will use a multimeter to measure the output voltage. If the sensor outputs a range of .1V, +/- .01V, linearly over the temperature range it passes. 2. The Sensor shall be able to reach into a cow's ear. a. We will use a tape measure to measure from the base of the sensor location on the board to the tip of the sensor. If the length is 150mm, +/- 10mm, it passes.

A.2 Power Module

Table A.2: Power Module Requirements and Verification

Requirement	Verification
1. The power module shall provide power to the device at least 2 years in temperatures that cows can survive outside, as low as 0 C[9] and up to 50C. a. The power module shall provide 20mAh total, +/- 1mAh, over at least 2 years. b. The power module shall be able to provide	1. The power module shall power the device for no less than 2 years. a. We will run the power module, with a 225 Ω load, for 4 hours and measure the total Amp-hours used with a ammeter. If the device uses less than 4.57 uAh, +/- .2 uAh, then the module passes. b. The power module, with a 225 ohm load, will be put in a temperature chamber that will

13.4mA, +/- .1mA, and a constant 3V, +/- .1V. in temperatures 0 C to 50C.	vary the temperature from 0C to 50C over 4 hours, rate of 12.5 C/hour, and power measurement, with a wattmeter, will be taken. If the power module outputs 13.4mA and a constant 3V, +/- .1V, it passes.

A.3 Control Module

Table A.3: Control Module Requirements and Verification

Requirement	Verification
1. The control module software shall receive sensor data. a. The control module software shall send an enable signal every 5 minutes, +/- 1 second. b. The control module software shall allow a 1ms, +/- .5ms, wake up time before each processing of sensor data. c. The control module software shall go into standby mode after transmission is finished. 2. The control module shall package the sensor data. a. The control module software shall convert the analog thermistor voltage data into 4 bytes of digital data and append the voltage data onto the 4 byte ID data to make an 8 byte packet. 3. The controller module shall hold in memory a unique ID. a. The controller module shall use memory to save a unique 4 byte ID.	1. The control module shall receive sensor data. a. Run the control module for 30 minutes and using an oscilloscope, track the input voltage and current. If we observe an enable signal every 5 minutes then it passes. b. Run the device for 30 minutes and using an oscilloscope, track the input voltage and current. If we observe a 1ms, +/- .5ms, active current draw, 2.5mA, before a transmitting current draw, 13.4mA, then it passes. c. Run the device for 30 minutes and using a oscilloscope track the current draw. If between the active/transmit cycle, every 5 minutes +/- 1 second, the current draw is the standby current, 0.7 µA, then it passes. 2. The control module shall package the sensor data. a. Run the control module for 30 minutes and use an oscilloscope to track the “tx” output pin. If every 5 minutes, +/- 1 second, the “tx” output pin outputs an 8 byte packet, with the first 4 bytes as the digital data and the last 4 bytes as the ID data, it passes. 3. The controller shall hold in memory a unique ID. a. Run the control module for 30

	minutes and use an oscilloscope to track the “tx” output pin. If every 5 minutes, +/- 1 second, the “tx” outputs and the first 4 bytes are the unique ID it passes.

A.4 Transmitter Module

Table A.4: Transmitter Module Requirements and Verification

Requirement	Verification
1. The transmitter module software shall transmit the packaged data. a. The transmitter module software shall transmit the 8 byte packet at the 915Mhz ISM band at 10dBm, +/- 1dBm for at least 100m. b. The antenna shall be able to transmit data at least 100m on the 915MHz ISM band at 10dBm, +/- 1dBm.	1. The transmitter module software shall transmit the packaged data. a. Run the transmitter for 30 minutes and use an MXA signal analyzer to read signals on the 915MHz band. If an 8 byte packet is detected it passes. b. Run the transmitter for 30 minutes 100 meters away from the MXA signal analyzer that is reading signals on the 915MHz band. If an 8 byte packet is detected it passes.

A.5 Receiver Module

Table A.5: Receiver Module Requirements and Verification

Requirement	Verification
1. The receiver module shall receive the packaged data. a. The antenna shall be able to receive data at least 100m on the 915MHz ISM band. 2. The receiver module software shall process the package into temperature and ID. a. The raspberry pi software shall process the 8 byte data into the ID data and Temperature data. 3. The receiver module software shall convert the 4 byte temperature data into a	1. The receiver module shall receive the packaged data. a. Run a RF signal generator and push a signal to 915MHz 100m away from the receiver. If the receiver antenna pin outputs the transmitted data it passes. 2. The receiver module shall process the package into temperature values. a. Send an 8 byte signal using a RF signal generator over the 915MHz band, and if the raspberry pi separates the signal into two 4 byte packages, first temperature then ID, it passes. 3. The receiver module software shall

temperature value. a. The raspberry pi software shall process the 4 byte temperature data into a temperature value.	convert the 4 byte temperature data into a temperature value. a. Input a 4 byte value into the raspberry pi and if the software converts the bytes into a temperature value it passes.