Project github: https://github.com/will7007/Sweet-Potato-Sensor

Summary

What has been done

1. **The BME280 temperature sensor has been successfully configured to work with the AtTiny85**, and the sensor has been tested with the cheap RF transmitters in a sweet potato warehouse with a max range of about 8 feet. **Using RFID or Bluetooth has been completely ruled out.**
2. Since 8 feet isn't ideal, a higher-quality 433 Mhz transmitter is required. We have tested a relatively high-power 433 Mhz LoRa radio module on both the Arduino Uno and AtTiny85 and found that the LoRa radio is significantly more stable than the cheaper modules at long ranges. This has proven to be true both with line-of-sight transmissions, transmissions through walls, and transmissions through a foot or two of sweet potatoes within a metal refrigerator. **The AtTiny85 successfully works with the 433 Mhz LoRa module at a range comparable to the highest bandwidth setting of the same transmitter when being used by an Arduino Uno.**

What needs to be done

1. Use both the BME280 and 433 Mhz LoRa modules on the same AtTiny85 at the same time.
    a. These devices normally cannot be used together without disabling the ability to program the AtTiny85 without a high-voltage programmer. To ensure that the AtTiny85 can be reused, **a high-voltage programmer may need to be purchased.**
2. Test out the 433 Mhz RFM69HCW transmitter, and try both the LoRa module and RFM69HCW module with "rubber duck" SMA antennas to see if range improves.
    a. 433 Mhz rubber duck (round black antenna) antennas and side-mount SMA connectors will need to be purchased. See the 433 Mhz LoRa module's Adafruit tutorial for more information.
3. Fine-tune the power efficiency of the AtTiny85 in regards both to sleep mode and transmission power.
4. Construct a system in which data packets can be forwarded if needed, and where a base station can parse information received, keep track of active sensors, and send collected data to a computer.
    a. This will likely be the most time consuming task out of all the elements on this list.

Project github: https://github.com/will7007/Sweet-Potato-Sensor

Component Information

433 Mhz RFM96W LoRa Radio Modules (best results)

- We had the best range with these modules from within a large amount of sweet potatoes and a metal refrigerator (see the antenna testing section).
    - Product link: https://www.adafruit.com/product/3073
- These modules are described in detail on Adafruit's website, but it is important to note that if you are using a 3V Arduino (like a 3V powered AtTiny85) then you will not necessarily need to buy the Adafruit breakout board.
    - The tutorial for both this module and a different module is here: https://learn.adafruit.com/adafruit-rfm69hcw-and-rfm96-rfm95-rfm98-lora-packet-padio-breakouts/overview
    - I got working Arduino Uno sending/receiving code from here: https://learn.adafruit.com/adafruit-rfm69hcw-and-rfm96-rfm95-rfm98-lora-packet-padio-breakouts/rfm9x-test
- Also note that to legally use these modules in America, you will need to be a licensed amateur radio operator. The one-way transmission of telemetry is allowed, provided you still follow regulations regarding callsigns (in my code, I have my callsign transmitted along with every message). Make sure to use your OWN callsign in your code.
- I used Sandeep Mistry's Library to get the LoRa modules to transmit from the AtTiny85 to the Arduino Uno (you must use matching sending/receiving libraries since the Adafruit library and Sandeep's use different packet structures).
    - **Note that you must SWITCH THE MISO AND MOSI PINS from where they would normally connect on the module from the AtTiny85 for the code to work. I am not sure why this is.**
- I did not try using the LoRa module in conjunction with the BME280 due to the combined use of both devices requiring the reset pin of the AtTiny85 to be turned into a digital output via the burning of an e-fuse, which cannot be easily reversed using normal means. Since I was able to get the BME280 running with the cheaper sending modules, I am confident that they should work together.
    - This project (link below) does something very similar, but it uses the encrypted The Things Network, a LoRaWan system. Note that not only will this not work for us as it does not connect two LoRa modules directly, but it also cannot be used legally as amateur radio operators cannot encrypt their transmissions.
      Link: https://www.thethingsnetwork.org/labs/story/tinylora-atmospheric-sensor
- As a cheaper alternative to this module (about half the price both in breakout board form and on Digikey), the 433 Mhz RFM69HCW (https://www.adafruit.com/product/3071) does not use the LoRa protocol but still is likely to be viable for our application (though I believe it will still require an amateur radio license).

Project github: https://github.com/will7007/Sweet-Potato-Sensor

- The LoRa module was able to outperform the cheaper 433/315 Mhz modules using a 6.5 inch straight wire. However, a rubber duck antenna/mount for the antenna can be purchased from adafruit to go with their breakout boards, which may increase range.
    - 433 Mhz Rubber Duck antenna: https://www.amazon.com/bestseller2016-433Mhz-Antenna-Straight-Shipping/dp/B072Q7319V
    - Connector for this antenna: https://www.amazon.com/Yootop-Female-Connectors-Center-Connector/dp/B07CMNDD3J/ref=sr_1_3?keywords=edge+launch+sma+connector&qid=1563997101&s=electronics&sr=1-3
    - These products are also on Adafruit if the Amazon links no longer work.
- The stickers which contain MAC addresses, included with the LoRa modules, are not tied to any specific module. The following was stated by an Adafruit employee: (https://forums.adafruit.com/viewtopic.php?t=128200#p638798)
    - “The slips are for convenience, and aren't associated with any specific board. LoRa radios need a UUID to connect to the radio-to-internet-to-radio LoRa network, but the radios we use in the Feathers don't have a hardware address. People have found that you can use any globally unique ID though, so we put a printed UUID in the bag with each board. If you aren't trying to connect to the LoRa network, you can ignore the UUIDs entirely. You don't need them for connections between the Feather and the computer, or for connections between any two radios. If you do want to connect to the LoRa network, you can use any UUID with any radio, just make sure each radio has a different UUID.”

Project github: https://github.com/will7007/Sweet-Potato-Sensor

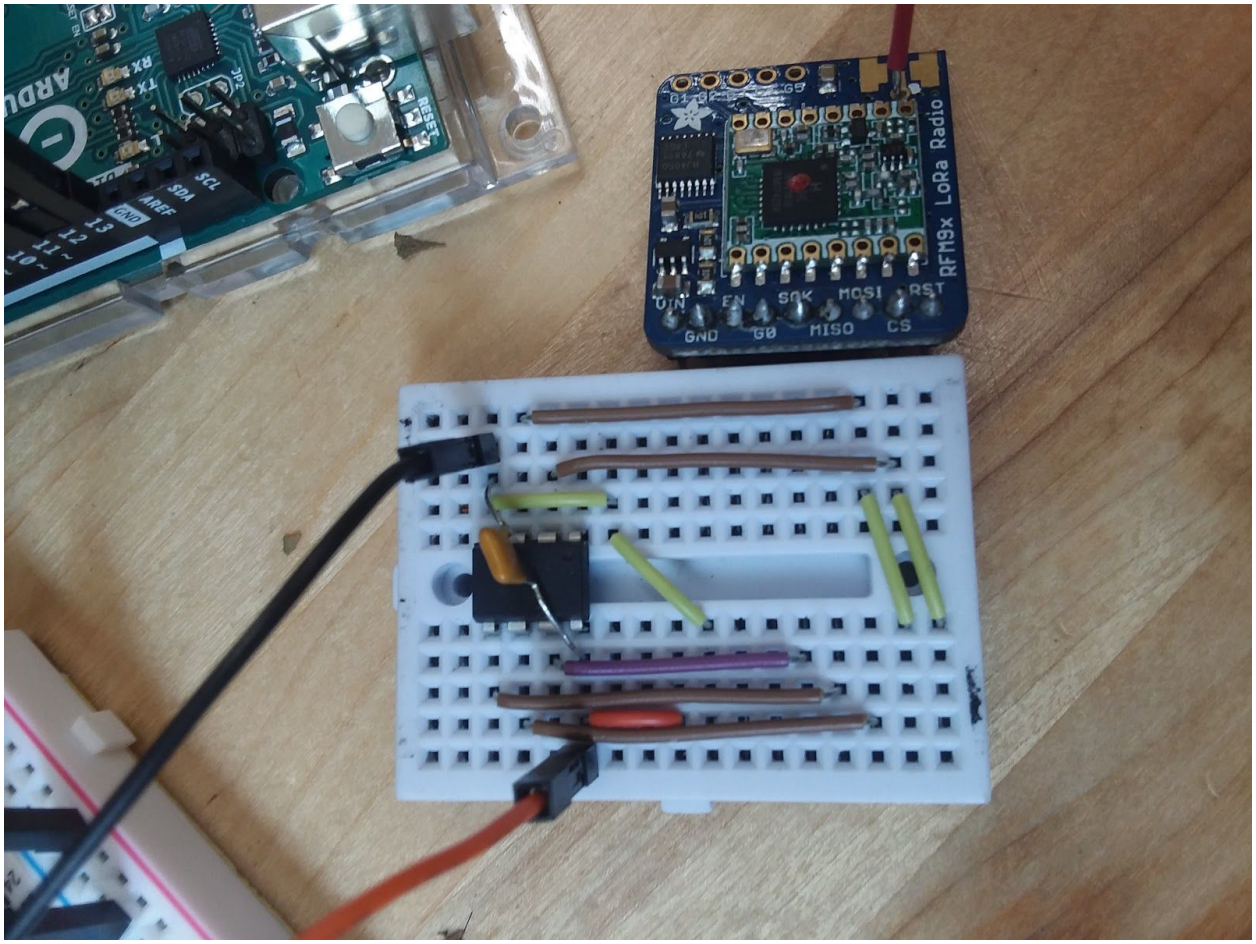

Image of the AtTiny85 Wiring for the LoRa Breakout Board (the breakout board is to be placed on the bottom row furthest to the right, with the RST pin on the last column of the board)
d0->MISO
d1->MOSI
d2->SCK
d3->CS (can be changed via code)
d4->RST (can be changed via code)

RF Tx/Rx 433/315 Mhz Modules

- Almost no documentation exists on these modules aside from a few lines of text describing their theoretical range and recommended voltage levels
    - Modules used:
    - Cheap 315 Mhz Modules: https://www.amazon.com/HiLetgo-Transmitter-Receiver-Arduino-Raspberry/dp/B00LNADJS6/ref=pd_sbs_147_1/133-9349170-2083021?_encoding=UTF8&pd_rd_i=B00LNADJS6&pd_rd_r=bddb100f-79c1-11e9-ab53-6922a0711ce1&pd_rd_w=7C3WU&pd_rd_wg=0LQpP&pf_rd_p=588939de-d3f8-42f1-a3d8-d556eae5797d&pf_rd_r=P43HZYXV41V126QEG9GJ&psc=1&refRID=P43HZYXV41V126QE

Project github: https://github.com/will7007/Sweet-Potato-Sensor

    - Cheap 433 Mhz Modules: https://www.amazon.com/HiLetgo-Wireless-Transmitter-Receiver-Raspberry/dp/B01DKC2EY4/ref=sr_1_4?keywords=433mhz+module&qid=1565367369&s=electronics&sr=1-4
- Put a 10k pull-up resistor on the data pin of the receiver (not required to make it work, but it will not have as much post-transmission noise)
    - The stability of the voltage source seems to have an effect on how effective the pull down resistor is. The Arduino uno would cause the pull-down resistor to not work as well, while on the NI ELVIS board it worked fine.
- I increased the number of pre-message tuning pulses sent by the Manchester encoding library in my code. It should be pretty self-evident where to change these values in the Manchester.h file--I didn't notice a huge difference in range but it should have made a difference.
- Optional low-pass filter values for the reception circuit: about 15kΩ and 0.022μF (not the absolute best values but it worked fine. I wasn't able to get a reliable frequency reading of the received data from the NI ELVIS scope so I just changed values until noise was filtered out). The filter is mainly needed if you want to remove false high values when using a level-triggered wake interrupt with the receiver, it isn't required for normal operation.
    - AtTiny85 Level-based Interrupts: https://thewanderingengineer.com/2014/08/11/pin-change-interrupts-on-attiny85/
- For improved range, use at least a 17.3 cm piece of straight wire on both the transmitter and the receiver (a ¼th wavelength antenna without a reflective ground plane). If this antenna is too long, then the size can be reduced to this smaller antenna (https://www.instructables.com/id/433-MHz-Coil-loaded-antenna/).
    - Note that this smaller antenna (referred to hereafter as the instructables antenna) only works well on the transmitter from my tests, but you are welcome to try and make designs of your own to test as my antennas were not perfect by any means.

BME280

- This is apparently the best temperature/humidity/pressure sensor, as reported by a variety of hobbyists
    - Link: https://www.adafruit.com/product/2652
- This chip also seems pretty simple to wire without the breakout board, although it is quite small
    - Node that there is a similar chip by the same company called the BMP280 that does not contain a humidity sensor which looks nearly identical. Chinese sellers may try to pass this off as the more expensive BME280 so caution should be used (at the time of writing, we have 2 breakout boards and 3 BME280 ICs so we shouldn't need to get more).

Project github: https://github.com/will7007/Sweet-Potato-Sensor

- While this chip works with ease on the “normal” Arduinos, I had some difficulty getting all the features to work on the AtTiny85 (most likely due to the extensive calculation process required to get human-readable data values). I was eventually able to read temperature and humidity (I didn’t try reading pressure since we don’t need it) with this library: https://www.14core.com/wiring-the-bme280-environmental-sensor-using-i2cspi-interface-with-microcontroller/. Note that it has a non-commercial license, but that shouldn’t be an issue if we aren’t actually selling this.

AtTiny85

- Library compatibility with the AtTiny85 has been the biggest thorn in the side of this project so far. During my time working on this project, I have gone through numerous libraries and switched cores (what lets the Arduino IDE know how to compile code for the AtTiny85) twice. These are the two cores I used:
    - David Mellis (https://github.com/damellis/attiny): I started off the project using this core. I do not remember the exact reason why I chose it--it could be as simple as it being the first working core I came across. I used it with the following libraries:
        - Manchester encoding library (https://github.com/mchr3k/arduino-libs-manchester) for use with the cheap 433/315 Mhz modules
        - The BME280 library listed in the BME280 section (www.14core.com)
        - TinySnore (https://github.com/connornishijima/TinySnore) to start timed sleep mode (which can be awoken from via a level change interrupt)
    - Spence Konde (https://github.com/SpenceKonde/ATTinyCore): This core seems to be more up to date/fully featured and should be used over the other core whenever possible.
        - I only used this core with Sandeep Mistry’s Library as Mellis’ core did not work.
        - I assume this library would work when compiling the BME280 library but I have not tried to compile it.
- Note that as a best practice, you should place a 100nF decoupling capacitor (included in the miscellaneous bag from my project, if that bag hasn’t been raided and disorganized by the time you read this in the far future) between the vcc and ground pins of the AtTiny85. The capacitor should ideally be as close as possible to reduce unwanted resistance/inductance but a few pins away on the breadboard shouldn’t be too bad.
- I used a 390µF capacitor between the reset and ground pins on an Arduino Mega to program the AtTiny85, but it’s likely that the value doesn’t matter too much.
    - Since guides exist for programming the AtTiny85 by using an Arduino Uno as an ISP, I won’t describe how to do it here.
- All the code in this project is designed to run at 1 Mhz. If you wish to change the frequency, be sure to burn the bootloader to the AtTiny85 again so your code doesn’t run 8 times/an eighth as fast.

Project github: https://github.com/will7007/Sweet-Potato-Sensor

RFID tag (ST25DV04K)

This chip wasn't used for the project, but I've documented how to use it both in this document and on my Github page.
We did not use these dynamic tags because they had very limited range even with antennas like this: https://www.iotrfidreader.com/sale-9377370-long-range-rfid-reader-hf-loop-antenna-high-frequency-13-56mhz-max-power-6w.html. The tag, and the protocol in general, does not support multi-hop transmissions, and the antennas are highly directional.

From Section 5.11:
Upon first receiving the chip;
Enabling the enabling of the mailbox: write 00000001b (a 1 in bit 0) to MB_MODE at address 000Dh. Device address E2=1. This will enable the mailbox buffer to be turned on. Note this address is in the I2C addressing space of the EEPROM, which can be read (but not written to) even when the mailbox is turned on.
Enabling the mailbox: write 1b to bit 0 of MB_CTRL_Dyn at address 2006h. Device address E2=0. Note that b7-b1 are read only so you will have to write just the one bit.

Making I2C wait for the RF to finish: the other version of this chip wouldn't let the I2C line write while an RF operation was going on, so I assume this chip is the same way. To avoid any data corruption in the buffer, configure the GPO to send an interrupt when there is RF activity by writing 10000010b to the GPO register at 0000h with the device address E2=1. This will overwrite the default configuration which uses the interrupt to let the I2C controller know when the field is entered or left.

Writing to the mailbox: the mailbox's address space is from 2008h to 2107h. The mailbox does not support overflow, but this shouldn't be a problem with only a few data points. The mailbox will raise up its flag every time a write is completed, and since the RF device can check the flag's status, we only need to write to the buffer when we have new data. When writing to the mailbox, the first address written to must be 2008h (the first bit of the mailbox).

Reading from the mailbox (RF): to check to see if the mailbox is up, send an RF read request command ADh at address 0Dh for bit 0, which will be 1 when the mailbox has its flag up. This is optional, as you can just read what's in the mailbox with the RF command Read Message, which is ACh. You can use this command to only read the address range in the mailbox that you know is filled with data.

Security session: the security session is enabled by default for the RF and I2C modes. The default password is 0000000000000000h (h for hex).

//Take off the LSB from the 8-bit I2C addresses in the datasheet for Wire.beginTransmission

Project github: https://github.com/will7007/Sweet-Potato-Sensor

//Do not use 0x[binary] for Wire.write, it will treat it as a hex number
//The tag will return 255 (1 for every bit) if you give it the system address when you should have given it the normal one

//I2C_SSO_Dyn 2004h page 55
//EH_CTRL_Dyn 2002h page 41

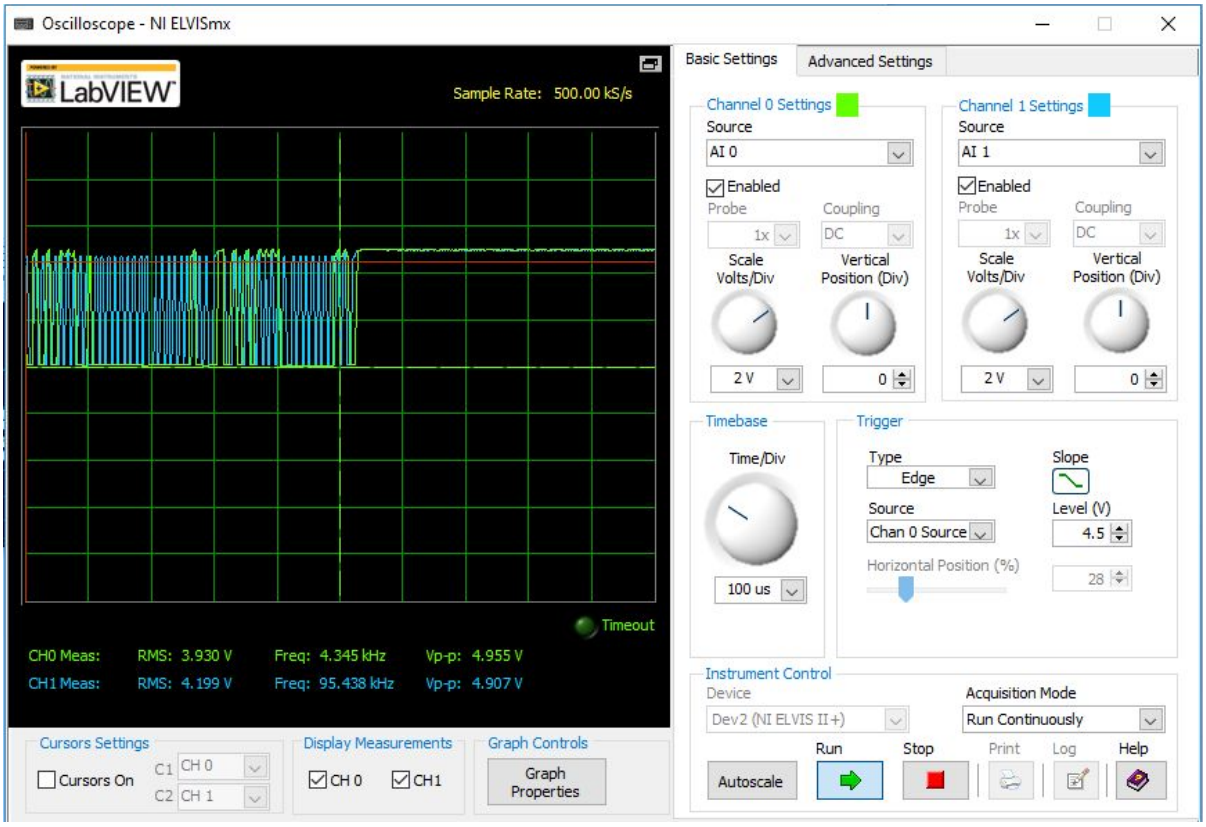

Figure: Reading the EH_MODE Register Via I2C

Project github: https://github.com/will7007/Sweet-Potato-Sensor

Tests

Cheap RF Module Antenna Tests

433 Mhz

For an antenna's distance to be valid, the receiver (the Arduino Mega) must have received 3 consecutive packets in order.
The transmitters were all lined up next to, but not touching, each other.
The temperature in the transmitting room was 72 F with 58% RH.
All tests were done with a ¼th wavelength receiving antenna on the Mega as this antenna worked the best in the sweet potato facility.
All but the ¼th wavelength antenna test at 5 V were done without the plastic cover on the Mega.

Antenna Type	Voltage (Volts)	Distance (location)	Approximate Distance (feet)
¼th Wavelength (~17.3 cm)	5	Outside silver doors of electronics lab	80
	~12 (11.71)	Outside room 170 (outside of electronics lab, a bit up the hall)	100
Amazon Antenna	5	On the second level of a metal cart (partial line of sight)	25
	~12 (11.71)	Under the fire alarm nearest the 194B door, on top of a wooden drawer	33
Instructables Antenna	5	In front of the floor 1 sign at the stairwell outside the silver doors of the electronics lab, on the intersection of the floor tiles	90
	~12 (11.70)	Outside room 170 (worked when the antenna connection was grounded with 10k resistor)	100

Repeater box was able to receive from near (not outside) the silver doors with bent ¼th antenna

Project github: https://github.com/will7007/Sweet-Potato-Sensor

315 Mhz

Valid range changed from 3 continuous packets to when the receiver gets 10 packets in a row

Sending Antenna Type	Receiving Antenna Type	Voltage (Volts)	Distance (location)	Approximate Distance (feet)
Instructables Antenna	¼th Wavelength (~23.8 cm)	5	Did not work with just 5 V, range seems to be highly dependent on voltage supplied	1
		~12 (11.70)	Inside of the conference room with the door closed, near the tv (note: signal seemed more stable overall while walking around/sending through walls vs 433 Mhz)	115
	Instructables Antenna	5	(after receiver fell on ground) Same as 5 V with ¼ th wavelength, was unable to get anything from far away	1
		~12 (11.70)	Inside of the conference room with the door closed, near the tv (was unable to receive in this exact point after the receiving arduino fell on the ground and bent the antenna)	115
Instructables Antenna	¼th Wavelength (~23.8 cm)	5	No range	0
		~12 (11.70)	No range, very erratic	0
	Instructables Antenna	5	No range	0
		~12 (11.70)	To wooden book bench near metal cabinet (line of sight)	30

The distance of the ¼ th wavelength antenna was about the same when the receiver's antenna was bent to be orthogonal to the receiver's ground plane, but the range seemed to be reduced when this happened to the transmitter.

Project github: https://github.com/will7007/Sweet-Potato-Sensor

Sweet Potato Test #1 (Scott Farms, Cheap Modules)

When using the instructables antenna with the receiver, we were not able to get more than 3 feet of range while the transmitter was near any sweet potatoes at all. After switching the receiving module to the 1/4th wavelength antenna, we were able to get about 8 feet of range when the module was beneath ~1 foot of sweet potatoes and had no problems getting 100+ feet in the open with line of sight (not near sweet potatoes).

The Cheap 433 Mhz Transmitter (grey box, left) and Two Bluetooth Modules in a sweet potato storage bin. The pallet bins are 84”x48”x36” and hold about 2000 pounds of sweet potatoes.

Project github: https://github.com/will7007/Sweet-Potato-Sensor

This is how much range I was able to get with the 1/4th wavelength antenna on the receiver and the indestructibles antenna on the sender (~4 feet). When using the indestructibles antenna on the receiver, I was only able to receive when pressed up against the edge of the pallet.

Project github: https://github.com/will7007/Sweet-Potato-Sensor

This is how far I was able to get with the same antenna setup as the previous picture (~8 feet of range). The transmitter is in the sweet potato container at the bottom-left corner of the image, underneath about a foot of sweet potatoes. The Bluetooth sensors that were also tested were completely prevented from sending underneath an identical load of sweet potatoes.

Sweet Potato Test #2 (Weaver Labs, LoRa Modules)

Under about two feet of sweet potatoes and within a metal refrigerator, the sensor was able to transmit about 75 feet on the most reliable bandwidth setting. When using the AtTiny85 (whose library does not seem to have the bandwidth as easily selectable) the range was about 50 feet--about the same as the Arduino Uno on its most speed-focused setting.

These photos combined show the total number of sweet potatoes (sans watermelons) that the LoRa radio was beneath.

Project github: https://github.com/will7007/Sweet-Potato-Sensor

The image on the left shows how the sensor looked after being covered up with some sweet potatoes, while the image on the right shows how far down into the container the transmitter was located (about in the middle of the white box). Note that the antenna is within the white tube--an orientation basically orthogonal to the receiver's antenna.

Project github: https://github.com/will7007/Sweet-Potato-Sensor

The image on the left shows how the sweet potatoes were stacked on the transmitter. No more sweet potatoes were added as there were no more left in the room.

Project github: https://github.com/will7007/Sweet-Potato-Sensor

This image was taken from the metal sweet potato refrigerator. The red cabinet on the left is 50 feet away, and the right cabinet is 75 feet away. The AtTiny85 and high-bandwidth Uno code were able to transmit to the left cabinet, while the lower-bandwidth Uno code was able to transmit to the right cabinet.

Done So Far (My To-Do List)

1. Order first set of components: Done May 17th
2. Make this document: Done May 20th
3. Finish reading the RFID tag datasheet and possibly email ____: Done May 20th
4. Optional: if the components still haven't arrived, then do a bit of the circuit schematic: Skipped, needed to do more research into RFID base stations and ISM bands
5. **Program the AtTiny with an example sketch: Done May 30th**
6. **Communicate with the temperature sensor: Done June 7th**
7. RF communication
    a. ~~Configure/find out a way to receive signals from the RFID tag using the dev kit:~~ RFID won't work for our use case
    b. **Configure the code to work with the 433Mhz Rx/Tx modules: June 4th**
    c. Make antennas for both solutions: June 17th
8. LoRa Radio:
    a. Pass ham radio technician exam to become licensed: July 22nd
    b. Try the LoRa radio on the Arduino Mega to obtain a performance baseline: 29th
    c. Get the LoRa module to send from the AtTiny85: August 1st
    d. **Final test with LoRa modules: August 2nd**
9. Document design process: August 9th

Notes Taken From Research Papers

Wireless Sensor Technology Review:

ISO15693 tags are mentioned but no research on them is presented

433Mhz transmitters dropped packets ~50% of the time while working with cattle

433Mhz transmitters worked better in wet conditions vs dry conditions

RFID tags at 13.56Mhz are noted for their short range (the paper says 20cm) and how it can take 5 seconds to send a temperature measurement

A 433Mhz transmitter worked the best vs 915Mhz and 2.4Ghz in a refrigerated marine container

Testing Network Protocols

Receiving messages is more energy-intensive vs sending them

Energy efficient protocols: reduce receiving time window (non applicable for wall-powered base station). Also, performing calculations consumes almost no energy at all vs

Project github: https://github.com/will7007/Sweet-Potato-Sensor

transmitting them so it will likely work out better if more calculations are performed on the AtTiny85

More efficiency: use the same radio frame (transmit an array of bytes for the manchester library we're using) to reduce radio uptime

TDMA

Put the temperature sensor in the center-top of the box for best results, but the corner of the box will reduce attenuation

RFID behavior study in enclosed trailer/container for real time temperature tracking (https://www.researchgate.net/publication/225752869_RFID_behavior_study_in_enclosed_marine_container_for_real_time_temperature_tracking)

Communication Technologies

DM2 and DASH7 specifications exist for the 433Mhz band

You should process the temperature data directly on the Arduino

You cannot avoid message forwarding and multi-hop (at least with 915 Mhz)

Project github: https://github.com/will7007/Sweet-Potato-Sensor

Daily Summary

I started to actually work on the project (after receiving all the parts) on May 30th and ended work on August 2nd, working about 4-6 hours a day on weekdays. Please contact me if you are interested in my day-to-day work schedule.