Josh Rabinowitz

“LOLIOT”

(Logging **O**f **L**ocal **I**nternal **O**ffice **T**emperature**)**

-Table of Contents
-Executive Summary

With the Internet of Things (IoT) quickly growing, manufacturers are pressured to release their products to market as soon as possible in order to keep up with competitors. Innovation and affordability are widely prioritized over basic security considerations as developers rush to get their designs into the hands of enterprises and consumers. Improving the state of IoT security in the future will depend on contributions from experts who understand the common pitfalls and their solutions.

The “LOLIOT” (Logging of Local Internal Office Temperature) project was conceived as an educational exercise to encourage consultants to go beyond simply hacking and think about IoT from the developer’s perspective, potentially honing new skills along the way. In addition to designing and building the system LOLIOT provides an opportunity for consultants to build upon the basic design and continue hardening and attacking with further iterations, either alone or with a group, to study how usability and functionality goals clash with security objectives.

Links:
https://mcuoneclipse.com/2014/08/09/hacking-the-teensy-v3-1-for-swd-debugging/
...

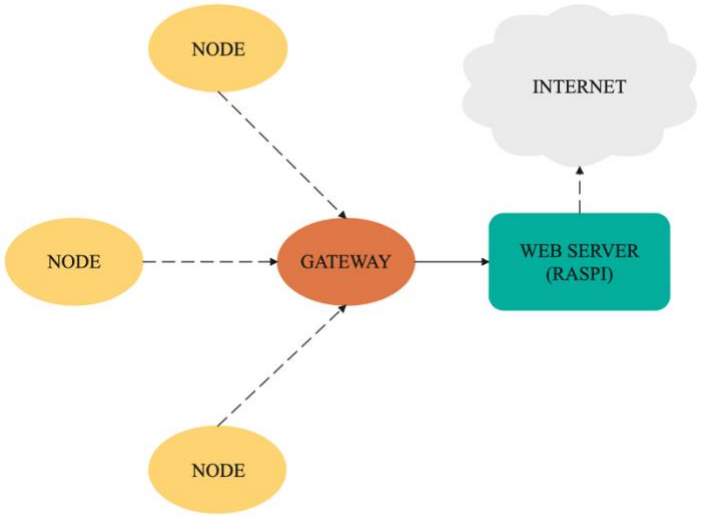
For each design component:
- -ask questions about security…
- -what are the threats?

-Purpose
- -Teaching new skillsets
  - -Circuitry
  - -PCB Design/ Soldering
  - -RF
  - -Microcontrollers
  - -Microcomputers
  - -Web Dev
  - -CAD/ 3d-printing??
- -Exercises
  - -Attack & Defend
    - -Blue team- adding layers of security
    - -Red team- finding attack vectors
  - -Augmenting design

-Switching to mesh topology
-Adding history/logging to web app
-Using different modules/controller

-Design

-Overview



-Protocols (Node):

[Temp. Sensor (*slave*)] **<= *I2C* =>** [Microcontroller (*master*)] **<= *SPI* =>** [RF Module (*slave*)]

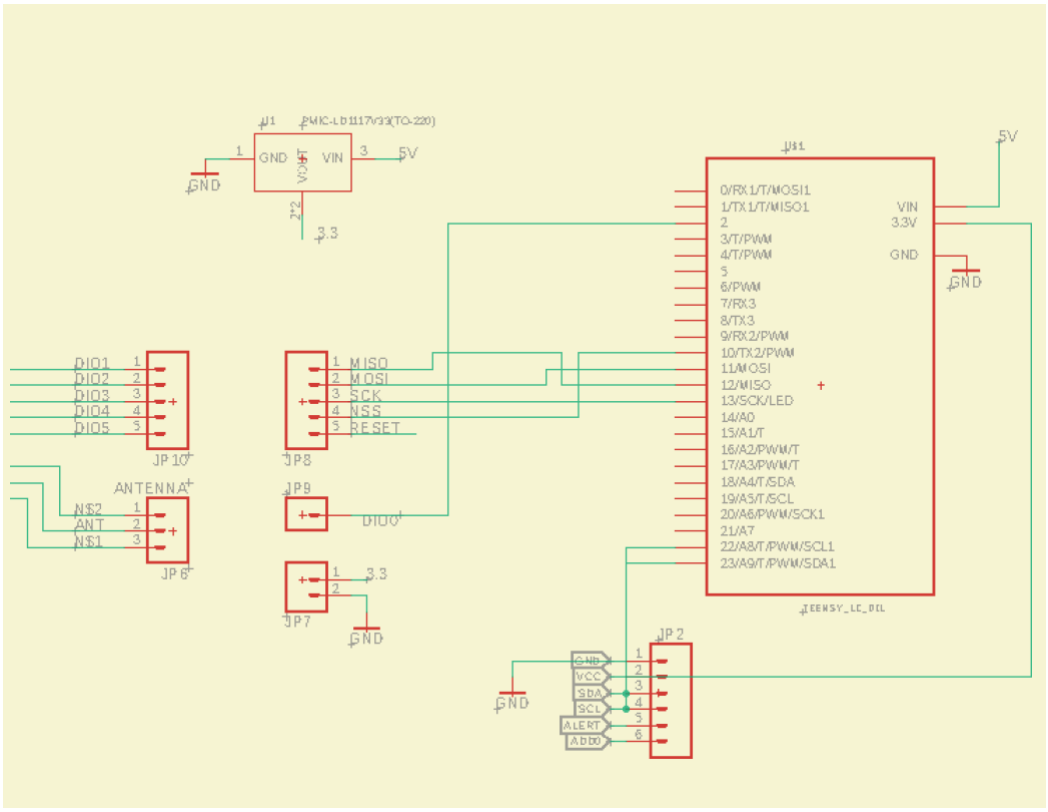
-Protocols (Gateway):

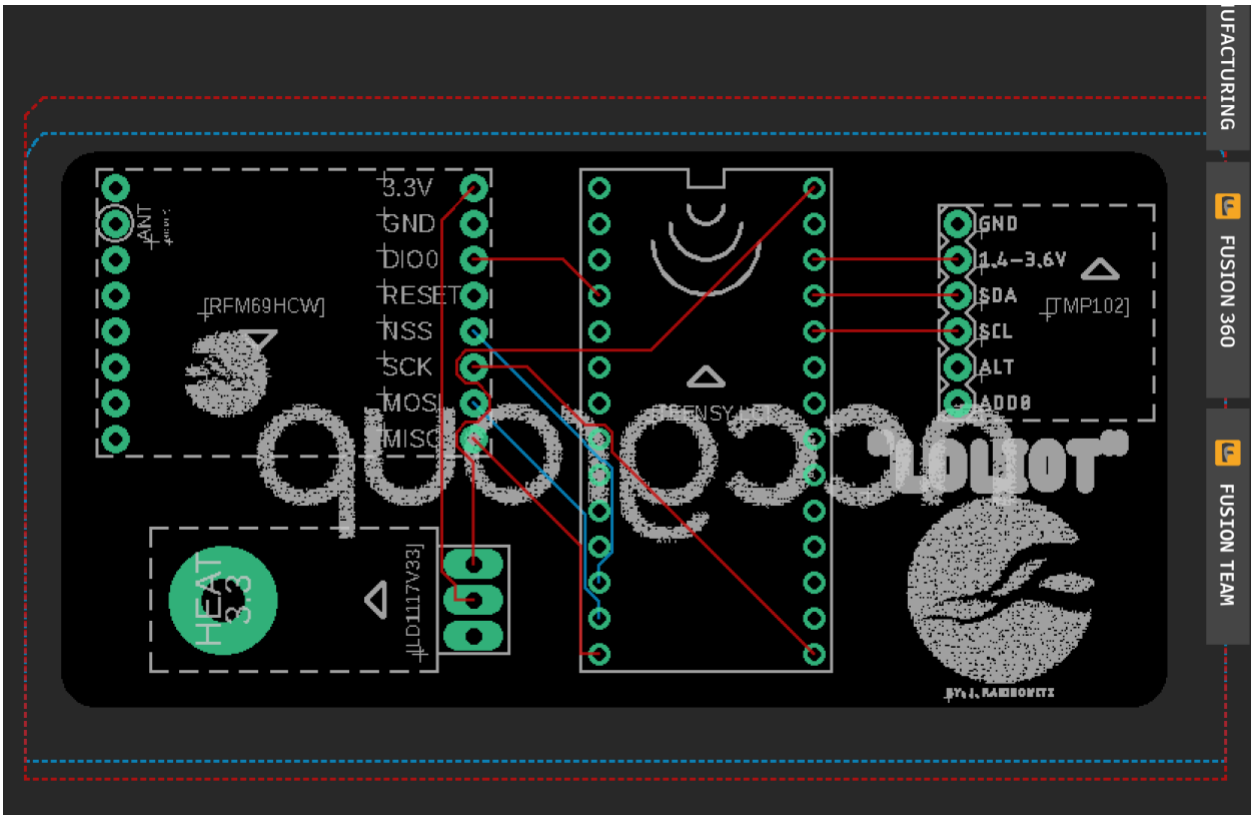
[RF Module (*slave*)] **<= *SPI* =>** [Microcontroller (*master*)] **<= *USB* =>** [Raspi/ Server]

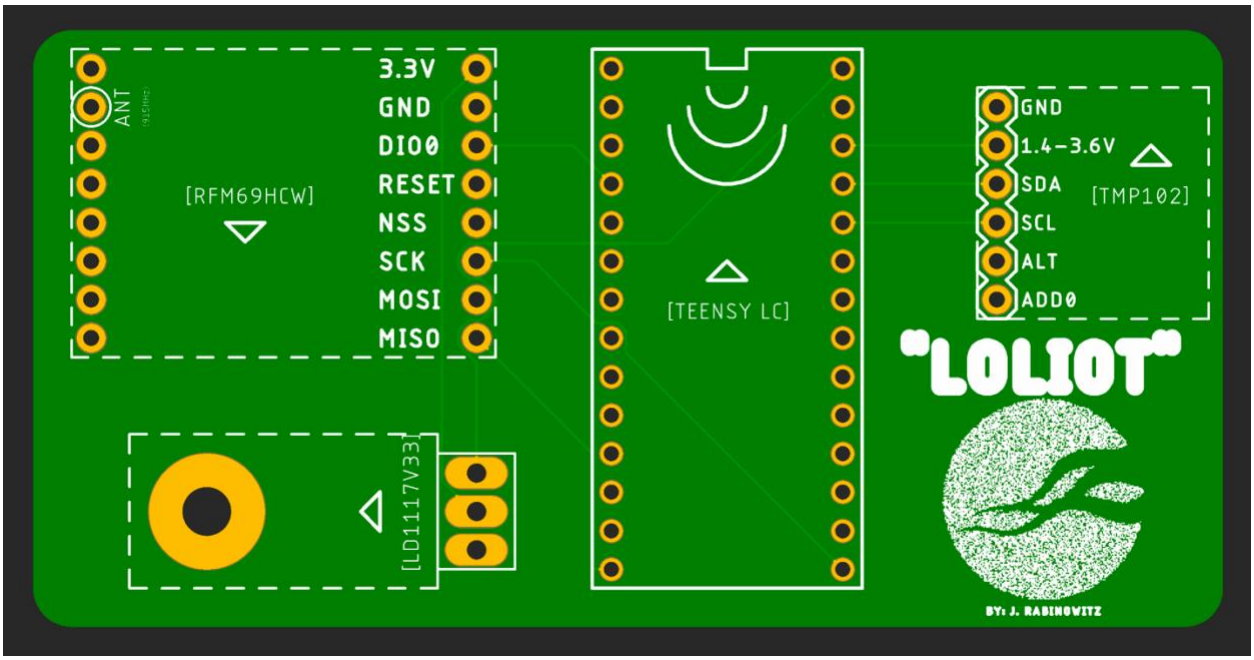
NODE:

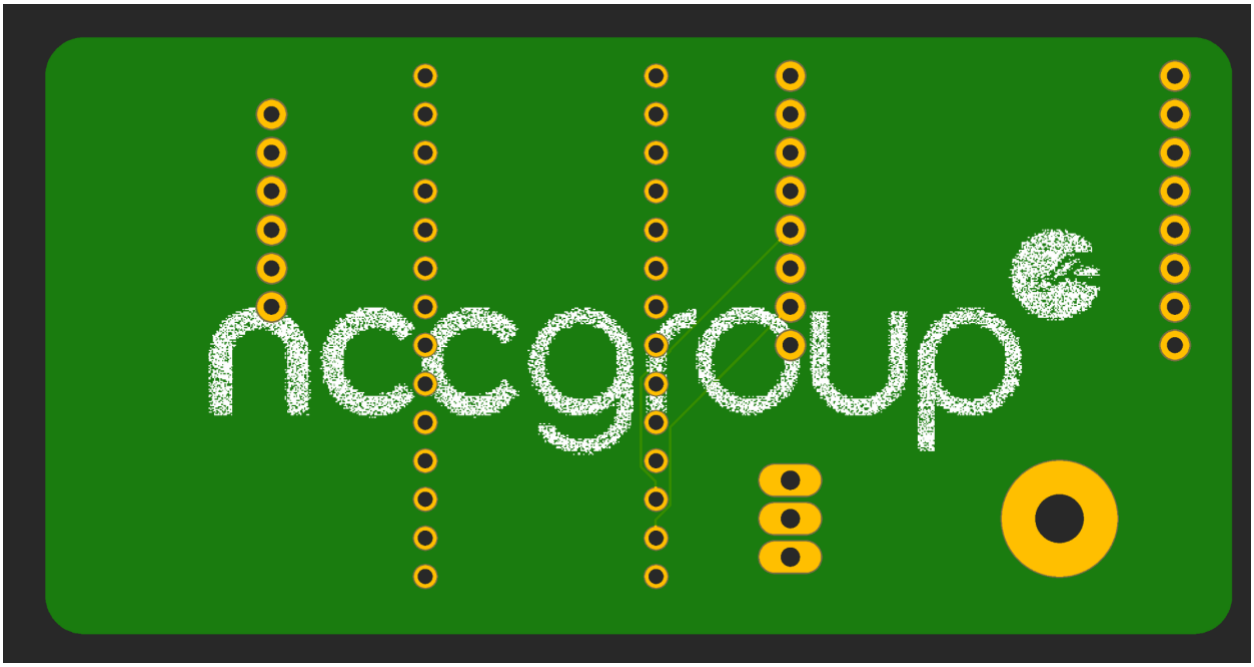
```
#include <SPI.h>
#include <RH_RF69.h>
#include <Wire.h>
```

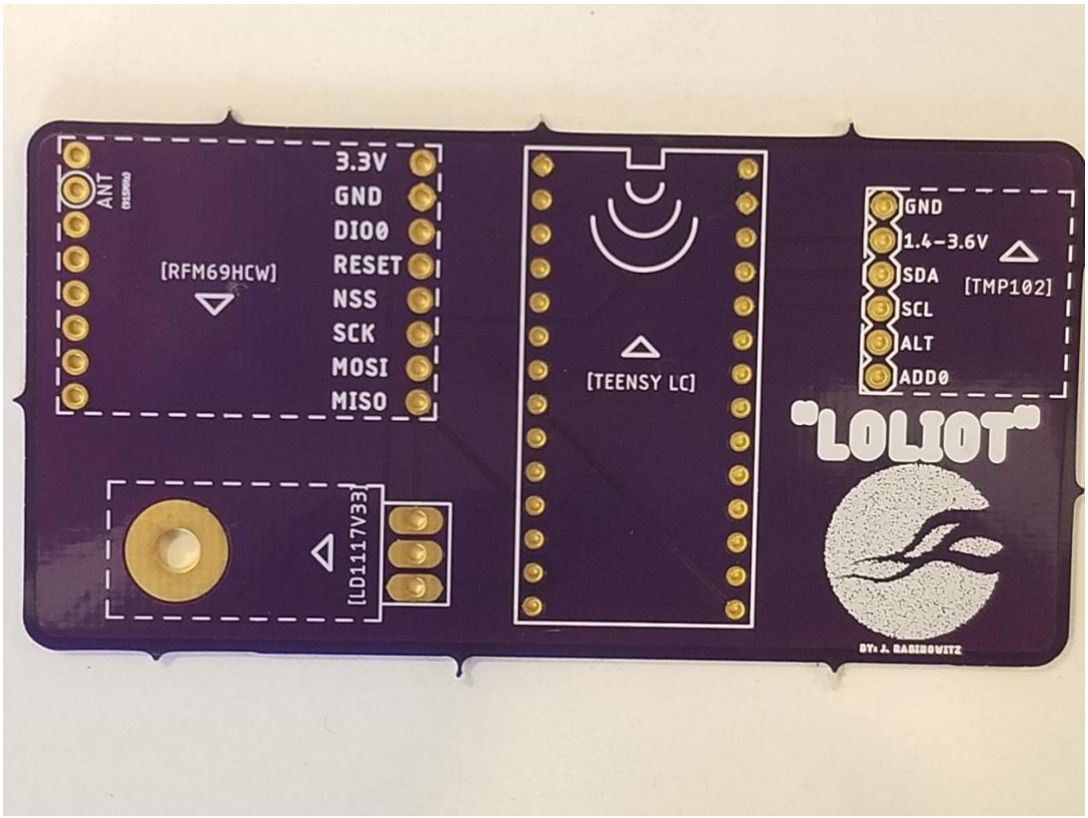
```
#include <SparkFunTMP102.h>

#define RFM69_CS  10
#define RFM69_INT 2
#define FREQUENCY 915.0
#define TRANSMIT_POWER 14 //(ranges from 14-20dBi)
#define KEY "encryptionkey123" // (must be 16 chars)
#define AREA "Kitchen" // (set a unique AREA for each node!)

//initiate driver
RH_RF69 rf69(RFM69_CS, RFM69_INT);
TMP102 sensor0(0x48);

void setup()
{
        Serial.begin(9600);
        // NOTE: remove if not tethered to computer:
        /*while (!Serial)
        {
                delay(1); //wait until console opens
        }*/
        sensor0.begin();

        if (!rf69.init())
                Serial.println("init failed");
        if (!rf69.setFrequency(FREQUENCY))
                Serial.println("setFrequency failed");
        rf69.setTxPower(TRANSMIT_POWER, true);
        //must match server's key:
        //rf69.setEncryptionKey(KEY);
}

void loop()
{
  //read temp
  sensor0.wakeup();
  float temperature = sensor0.readTempF();
  sensor0.sleep();

  //convert to data
  String tempStr = String(temperature);
  String mid = ", ";
  String fullStr = AREA+mid+tempStr;
  uint8_t data[sizeof(fullStr)];
  fullStr.getBytes(data, sizeof(data));

  //send to gateway
  if (rf69.send(data, sizeof(data)))
    Serial.println("[+] sending temp...");
  else
    Serial.println("[!] send failed!");
  delay(1000);
}
```


U1
MIC-LD1117V33(TO-220)
GND
VOUT
VIN
3
5V
3.3
U$1
0/RX1/T/MOSI1
1/TX1/T/MISO1
2
3/T/PWM
4/T/PWM
5
6/PWM
7/RX3
8/TX3
9/RX2/PWM
10/TX2/PWM
11/MOSI
12/MISO
13/SCK/LED
14/A0
15/A1/T
16/A2/PWM/T
17/A3/PWM/T
18/A4/T/SDA
19/A5/T/SCL
20/A6/PWM/SCK1
21/A7
22/A8/T/PWM/SCL1
23/A9/T/PWM/SDA1
VIN
3.3V
GND
TEENSY_LC_DIL
DIO1
DIO2
DIO3
DIO4
DIO5
JP10
ANTENNA
N$2
ANT
N$1
JP6
JP8
MISO
MOSI
SCK
NSS
RESET
JP9
DIO0
JP7
JP2
SDA
SCL
ALERT
ADD0

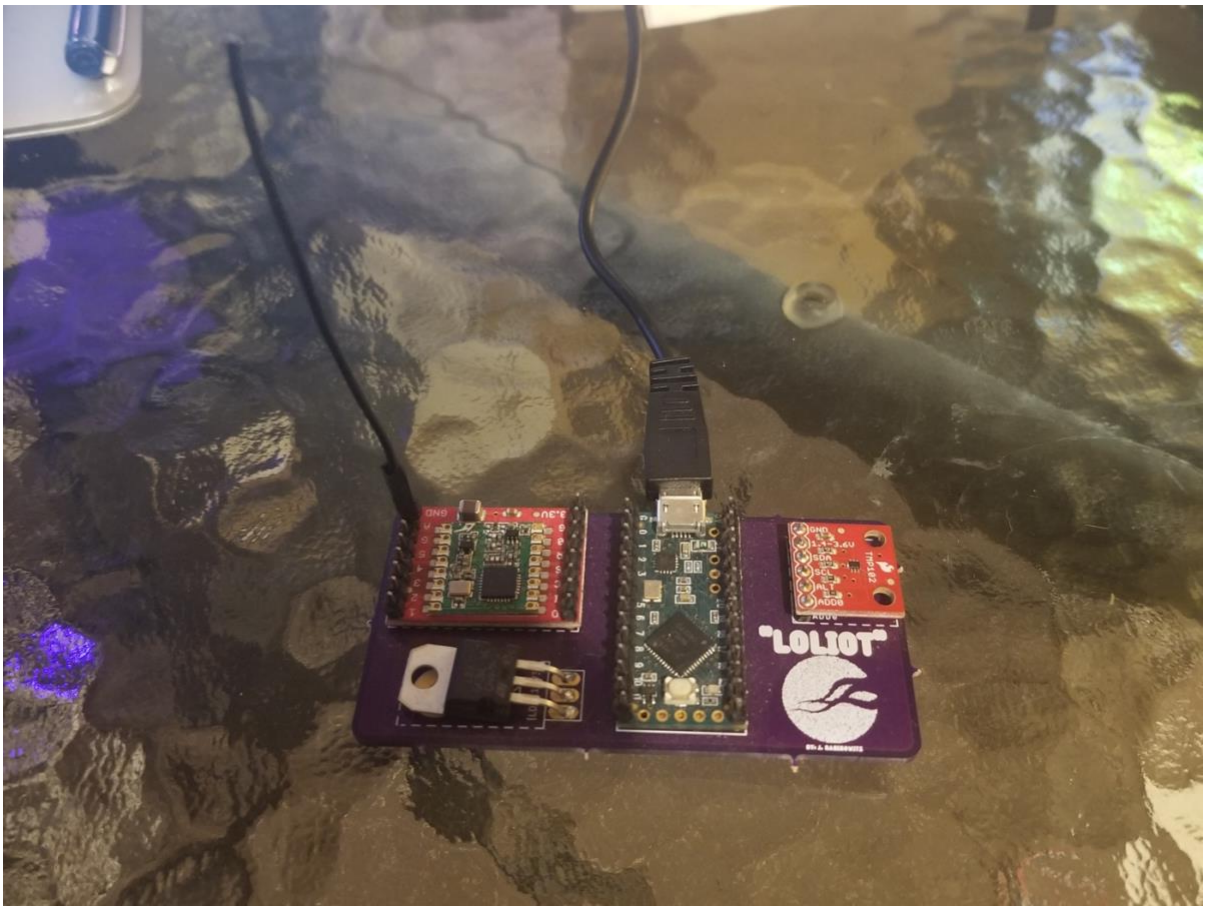

ANT
[RFM69HCW]
3.3V
GND
DIO0
RESET
NSS
SCK
MOSI
MISO
HEAT
3.3
[LD1117V33]
GND
1.4-3.6V
SDA
SCL
ALT
ADD0
[TMP102]
FUSION 360
FUSION TEAM


ANT
[RFM69HCW]
3.3V
GND
DIO0
RESET
NSS
SCK
MOSI
MISO
[LD1117V33]
[TEENSY LC]
GND
1.4-3.6V
SDA
SCL
ALT
ADD0
[TMP102]
"LOLIOT"
BY: J. RADINOWITZ


nccgroup

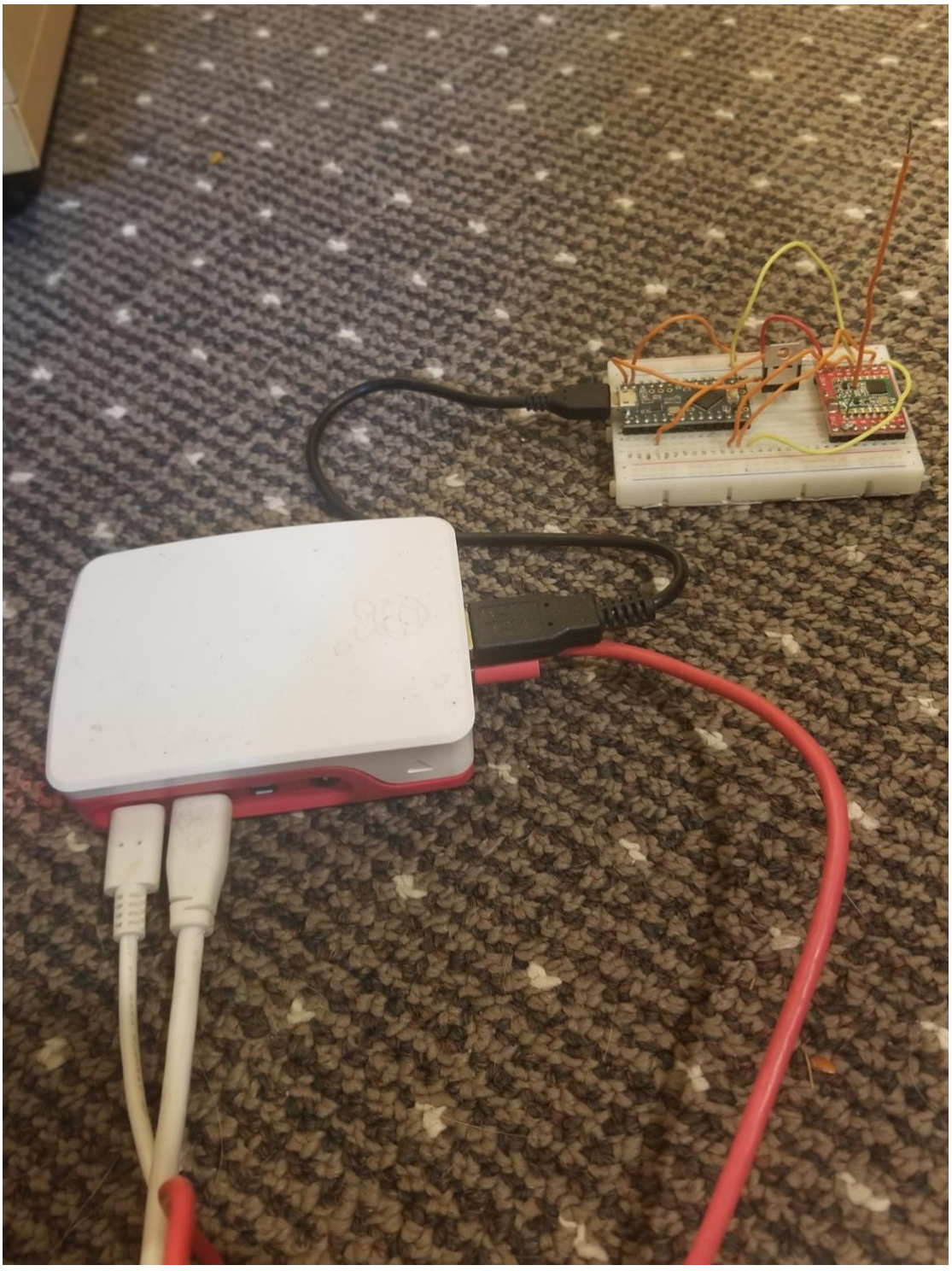

ANT
[RFM69HCW]
3.3V
GND
DIO0
RESET
NSS
SCK
MOSI
MISO
[LD1117V33]
[TEENSY LC]
GND
1.4-3.6V
SDA
SCL
ALT
ADD0
[TMP102]
"LOLIOT"
BY: J. RADINOWITZ


nccgroup


GND
3.3V
GND
3.3V
TMP102
SCL
SDA
ADDR
LoLiOT

GATEWAY:

```
#include <SPI.h>
#include <RH_RF69.h>

#define RFM69_CS  10
#define RFM69_INT 2
#define FREQUENCY 915.0
#define TRANSMIT_POWER 14 //(ranges from 14-20dBi)
#define KEY "encryptionkey123" //(must be 16 chars and match all Nodes)

//initiate driver
RH_RF69 rf69(RFM69_CS, RFM69_INT);

void setup()
{
  Serial.begin(9600);
  // NOTE: remove if not tethered to computer:
  while (!Serial)
  {
    delay(1); //wait until console opens
  }


  if (!rf69.init())
    Serial.println("init failed");
  if (!rf69.setFrequency(FREQUENCY))
    Serial.println("setFrequency failed");
  rf69.setTxPower(TRANSMIT_POWER);
  //must match server's key:
  //rf69.setEncryptionKey(KEY);
}

void loop() {
  Serial.println("Listening...");
  if (rf69.available())
  {
     // Should be a message for us now
    uint8_t buf[RH_RF69_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf69.recv(buf, &len))
    {
      Serial.print("got request: ");
      Serial.print((char*)buf);
    }
    else
    {
      Serial.println("recv failed");
    }
  }
}
```



RASPI (web server):
-templates
-'index.html'

```
<!DOCTYPE html>
   <head>
      <title>{{ title }}</title>
      <link rel="stylesheet" href="../static/style.css">
      <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
   </head>
   <body>
      <h1>{{ title }}</h1>
      <table>
         {% for key, value in data.items() %}
            <tr>
               <th> {{ key }} </th>
               <td> {{ value }} </td>
            </tr>
         {% endfor %}
      </table>
      <footer>Date/Time: {{ time }}</footer>
   </body>
</html>
```

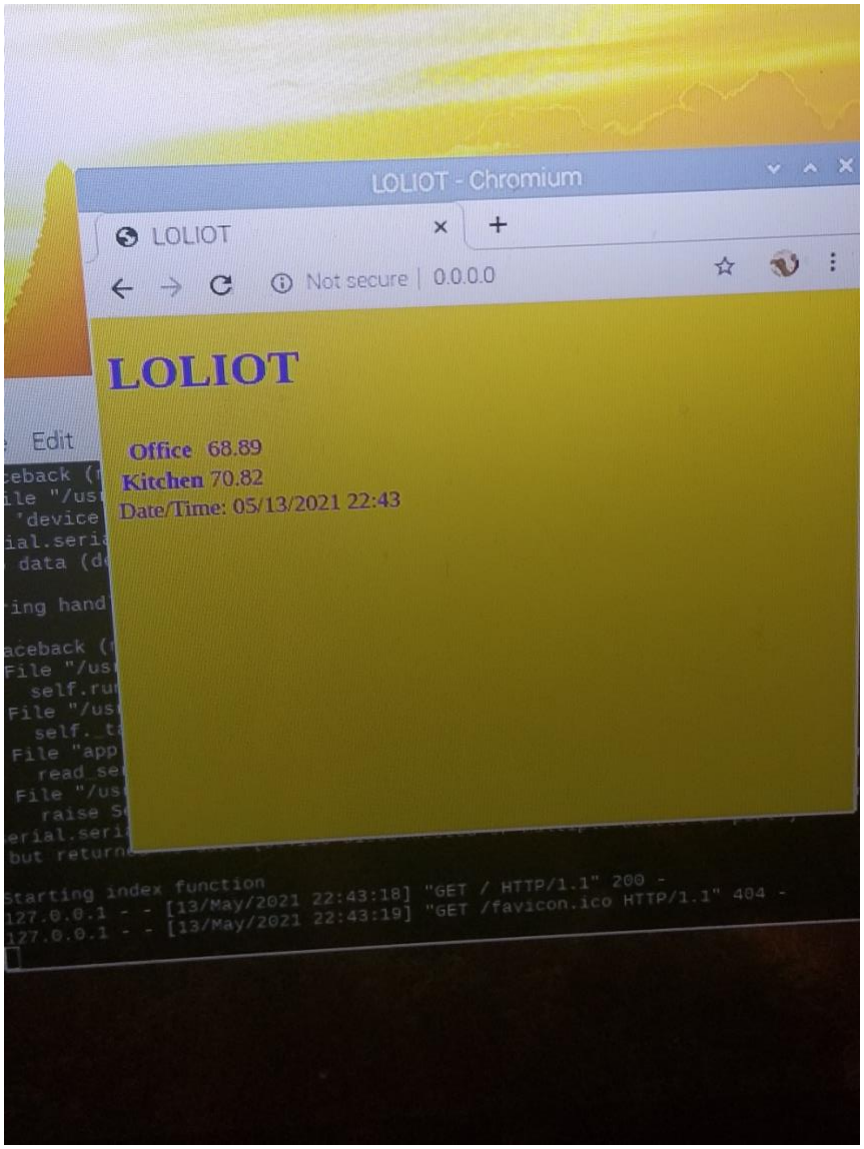
-static
-'style.css'

```
body {
  background: orange;
  color: blue;
}
```

-'app.py'

```
from flask import Flask, render_template
import datetime
import threading
import serial

data = ""
dict = {}
def listen():
    ser = serial.Serial('/dev/ttyACM0', 9600)
    print("Starting listen function")
    while True:
        read_serial = ser.readline().decode()
        data = read_serial.split('\n')[0]
        location = data.split(', ')[0]
        temp = data.split(', ')[1]
        dict[location] = temp

app = Flask(__name__)

@app.route("/")
def index():
    print("Starting index function")
    now = datetime.datetime.now()
```

```
    timeString = now.strftime("%m/%d/%Y %H:%M")
    templateData = {
        'title' : 'LOLIOT',
        'data' : dict,
        'time' : timeString
    }
    return render_template('index.html', **templateData)

if __name__ == "__main__":
    radio_thread = threading.Thread(target=listen, daemon=True)
    radio_thread.start()
app.run(debug=True, port=80, host='0.0.0.0')
```



*(*note: example above using only 2 active nodes- can add tons more!)*

-Security

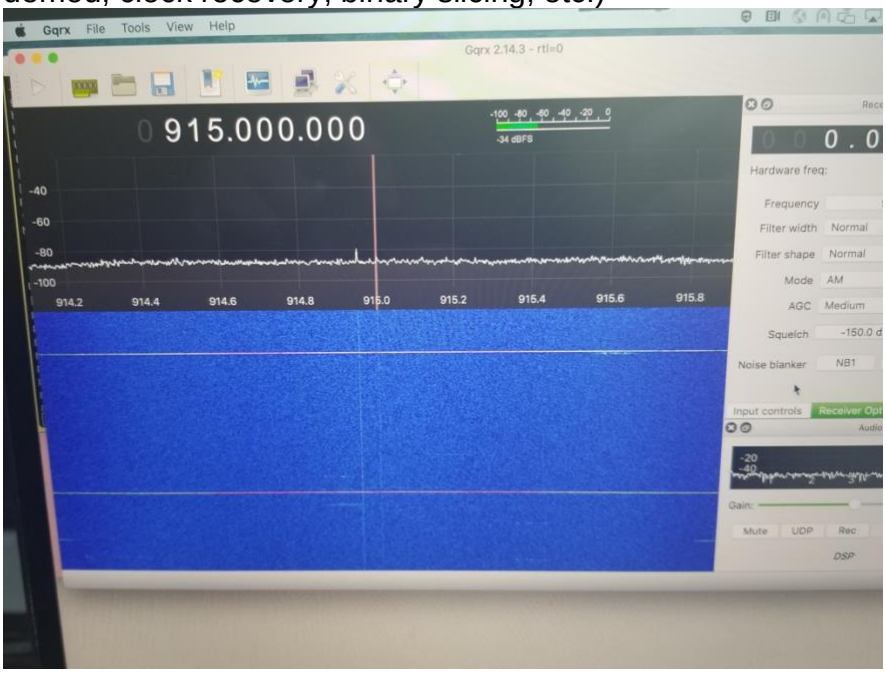
-Threat modeling/ Risks

-Hardware

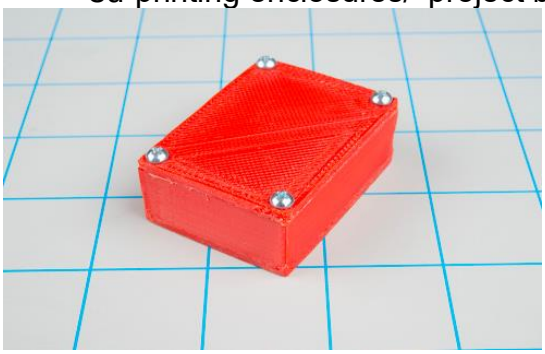
-Teensy LC
-Raspi
-Physical
-sd card
-Remote
-RF

(using SDR dongle/antenna to intercept signal from active node:)

(node signal was captured using GQRX- I/Q data can then be decoded via DSP to reveal raw transmission- this would be done using GNU Radio pipeline w/filtering, demod, clock recovery, binary slicing, etc.)



- -Web App
- -Defenses/ Mitigations
  - -Hardware
    - -Teensy LC
    - -Raspi
      - -Physical
      - -Remote
  - -RF
  - -Web App
- -Further work/research
  - -3d-printing enclosures/ ‘project boxes’



  - -Modifying the teensy for debugging
    - -(Include relevant links)
  - -Securing the raspberry pi/ web server
    - -Login/SSH
    - -Encryption
    - -Etc.
  - -Securing the web app
    - -Login
    - -Etc.
- -Appendix: Materials & budget
- -Appendix: Relevant links