ORACLE®

Howdy, I'm Blaine Carter

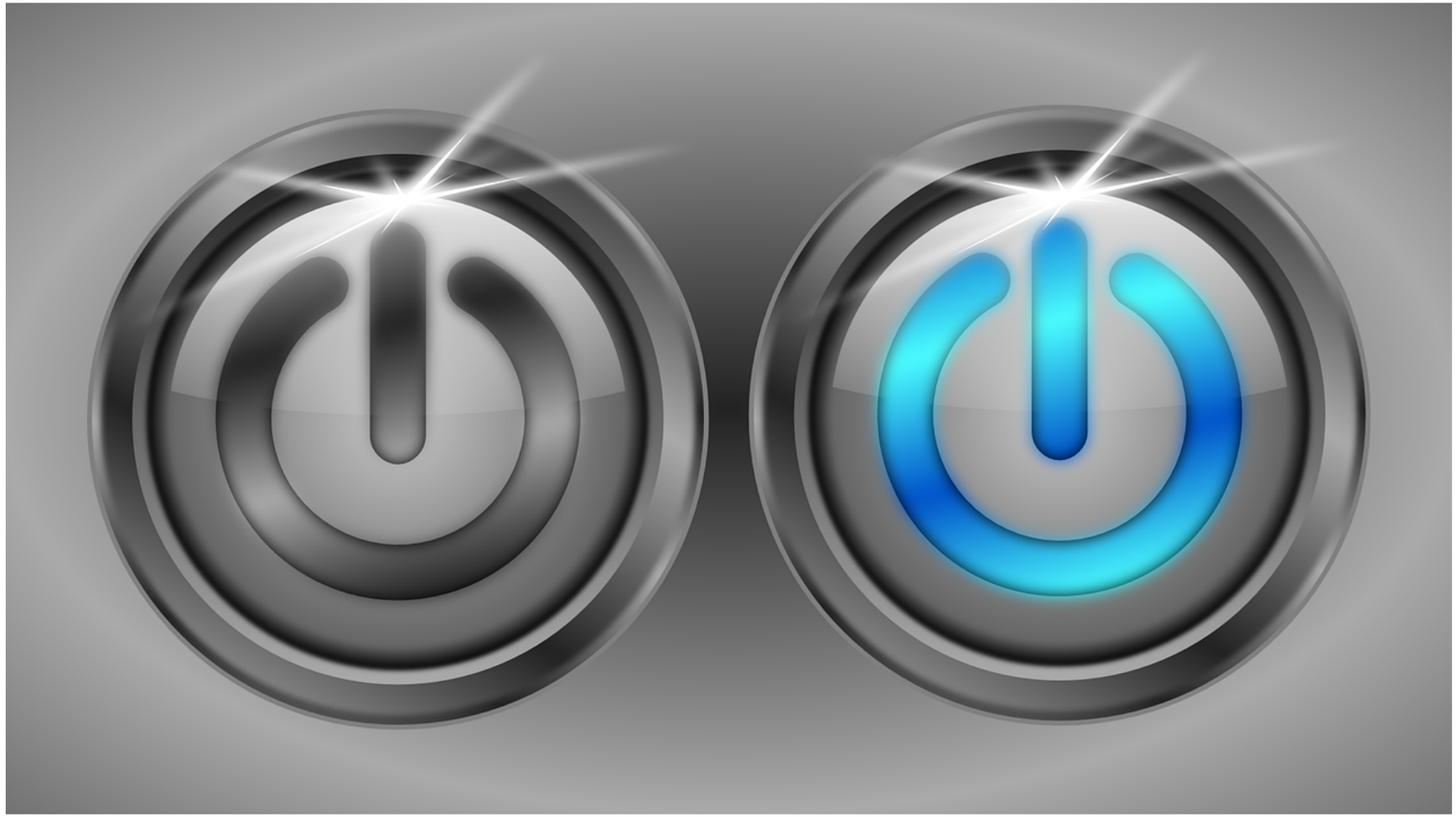
Oracle Corporation
Developer Advocate for Open Source

Email: blaine.carter@oracle.com
Blog: learncodeshare.net
Twitter: @OraBlaineOS
YouTube: www.youtube.com/blainecarter
Team: community.oracle.com/docs/DOC-917690

My Power Bill

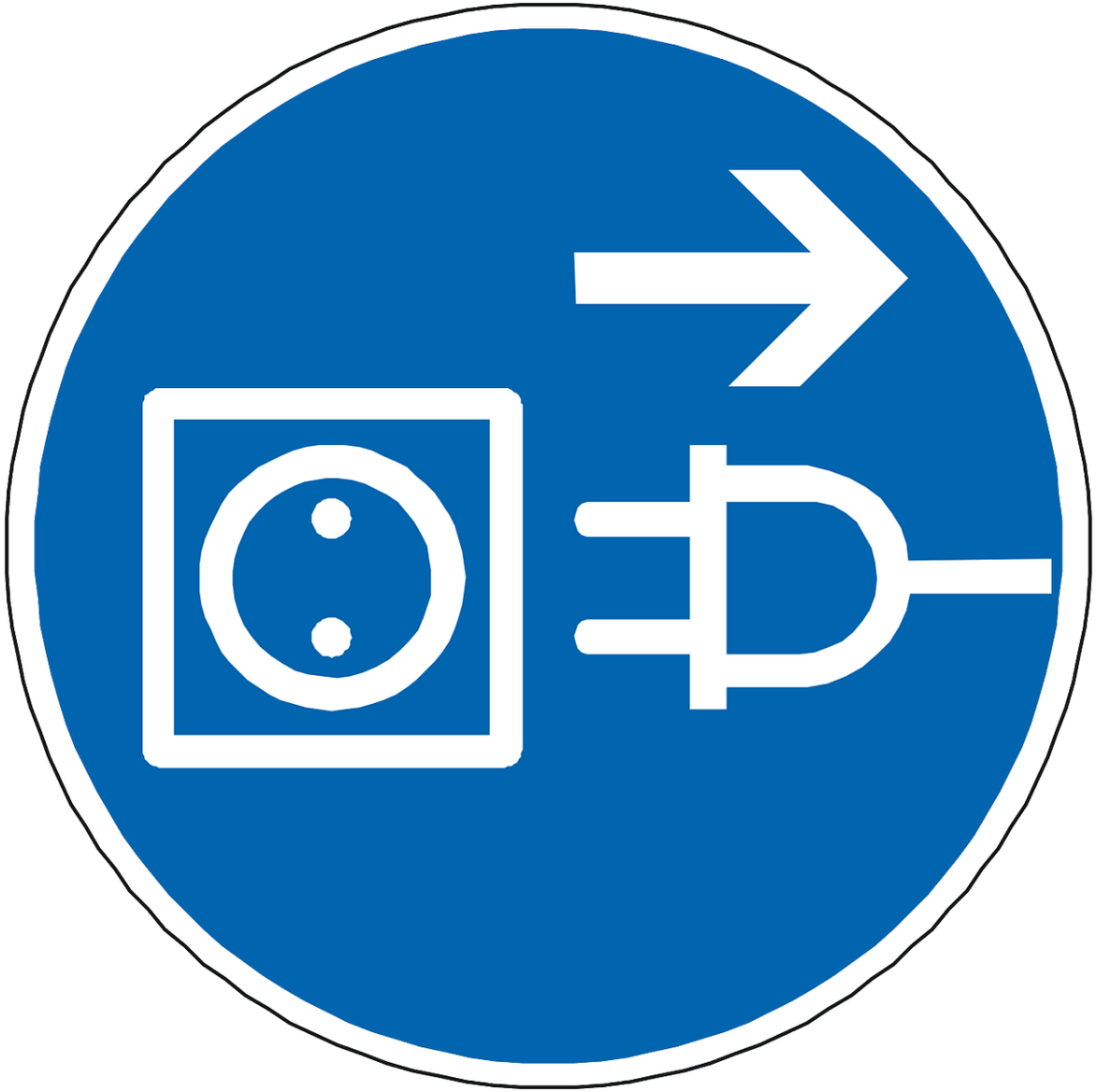
It is partially a ‘green’ thing

Use Less Energy

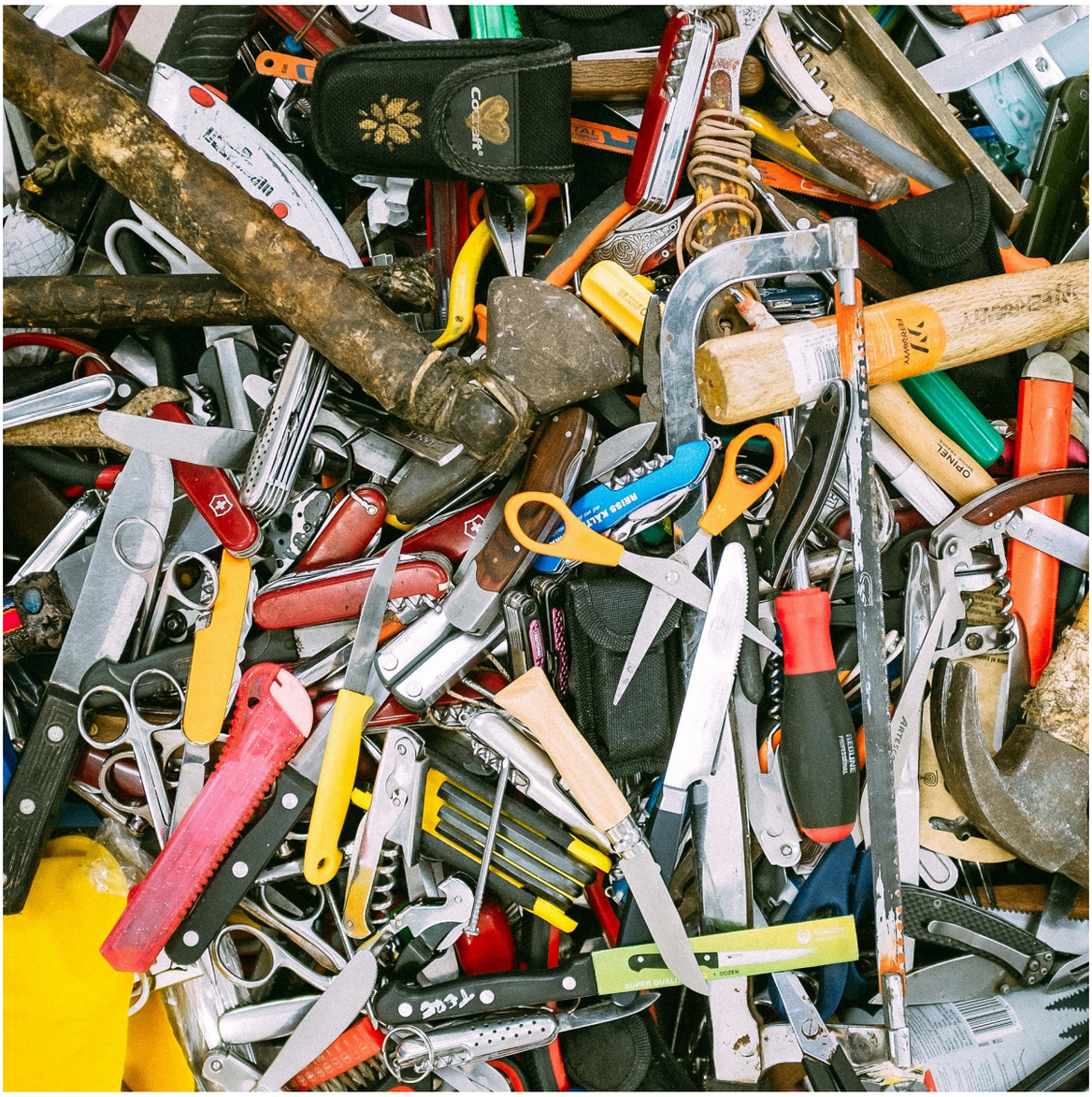


Kill-a-Watt

KILL A WATT®
METER CLOCK
120.3
Volt
Volt
Amp
Watt
Hz
KWH
VA
PF
Hour
G

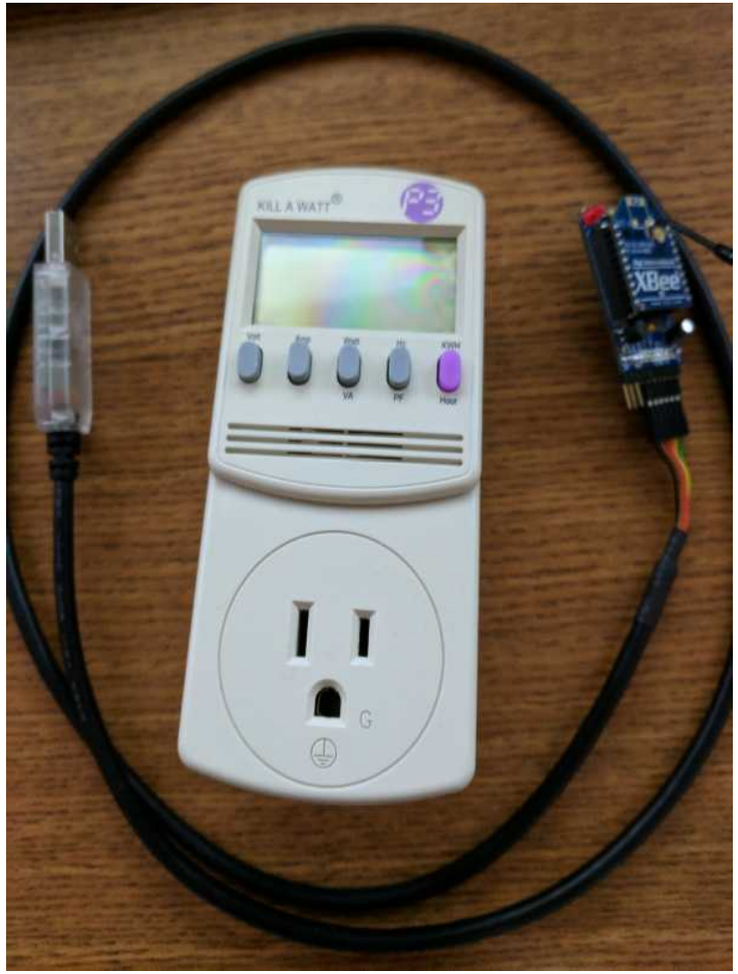


Keep track of the data

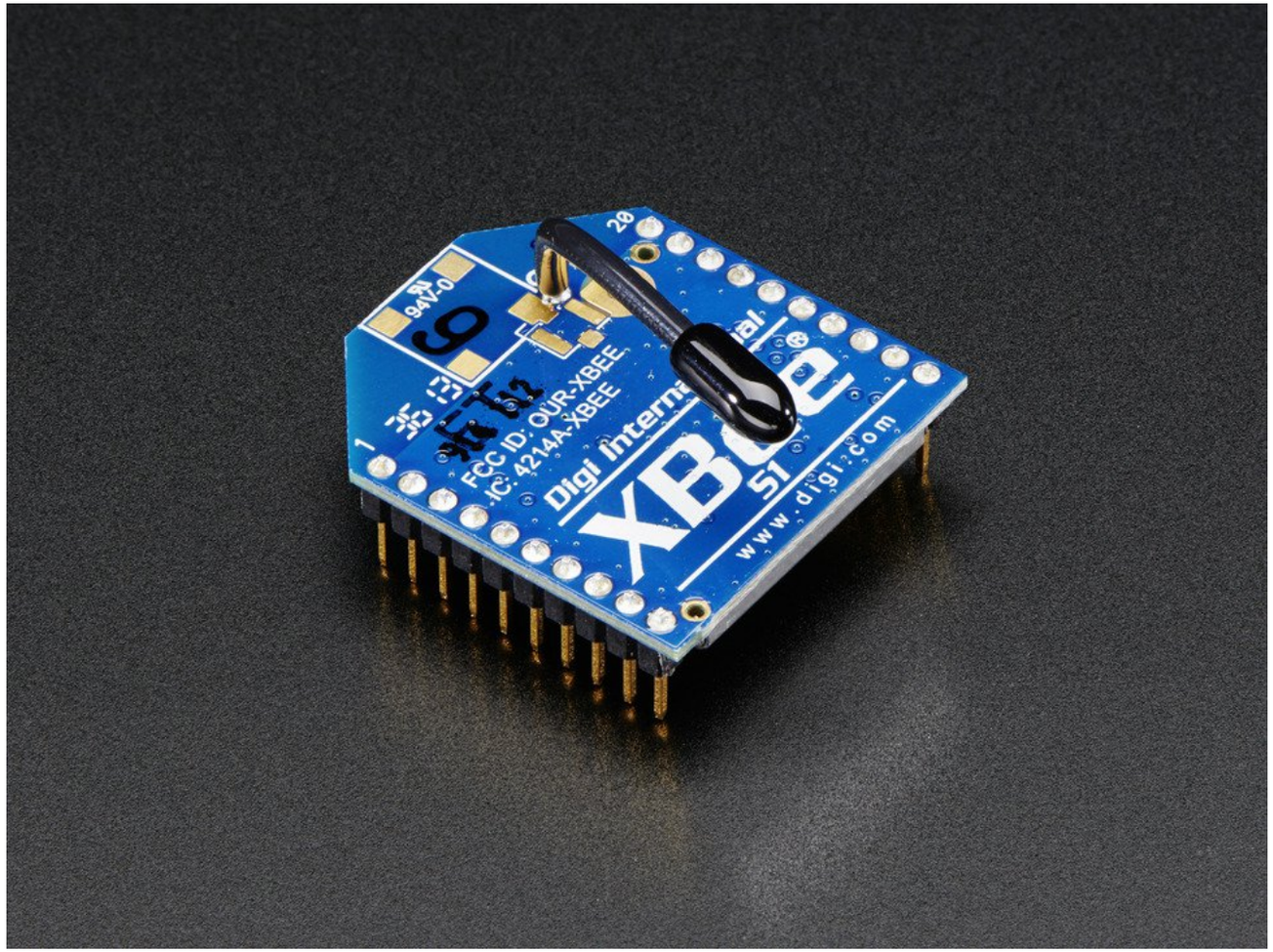


Tweet-a-Watt

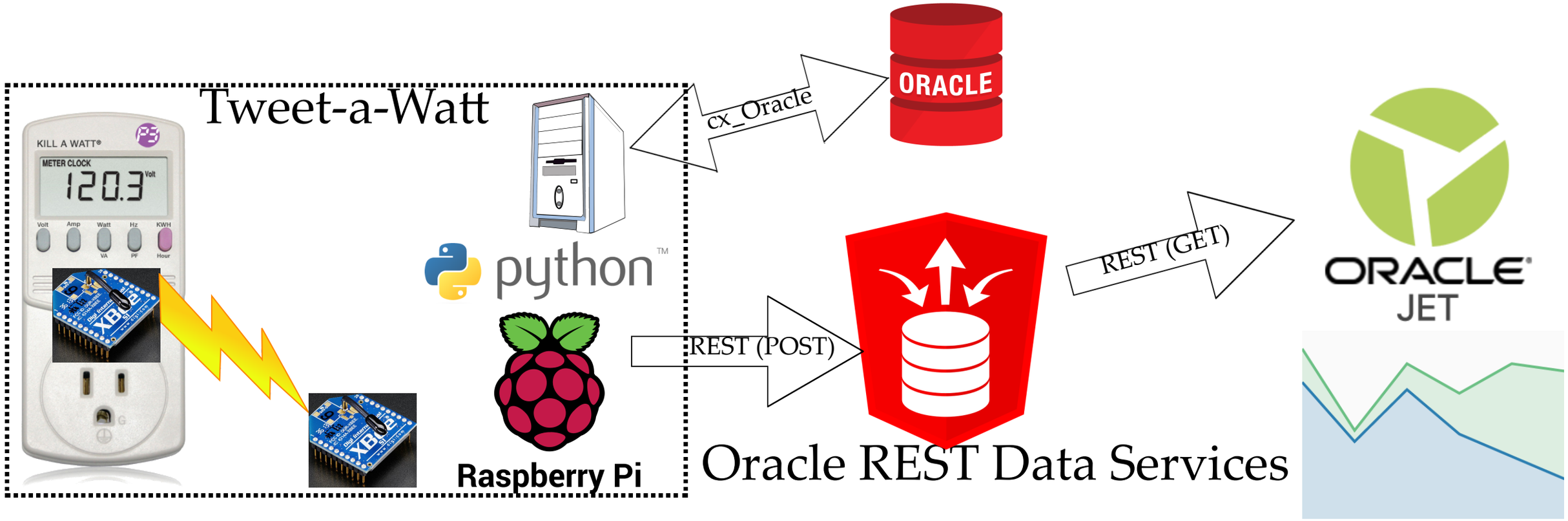
Lady Ada - https://learn.adafruit.com/tweet-a-watt?view=all



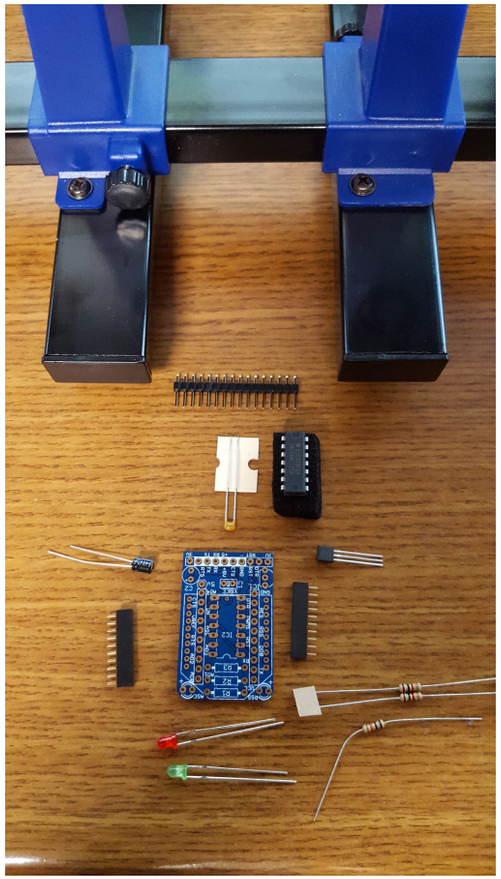
X-Bee



Overview

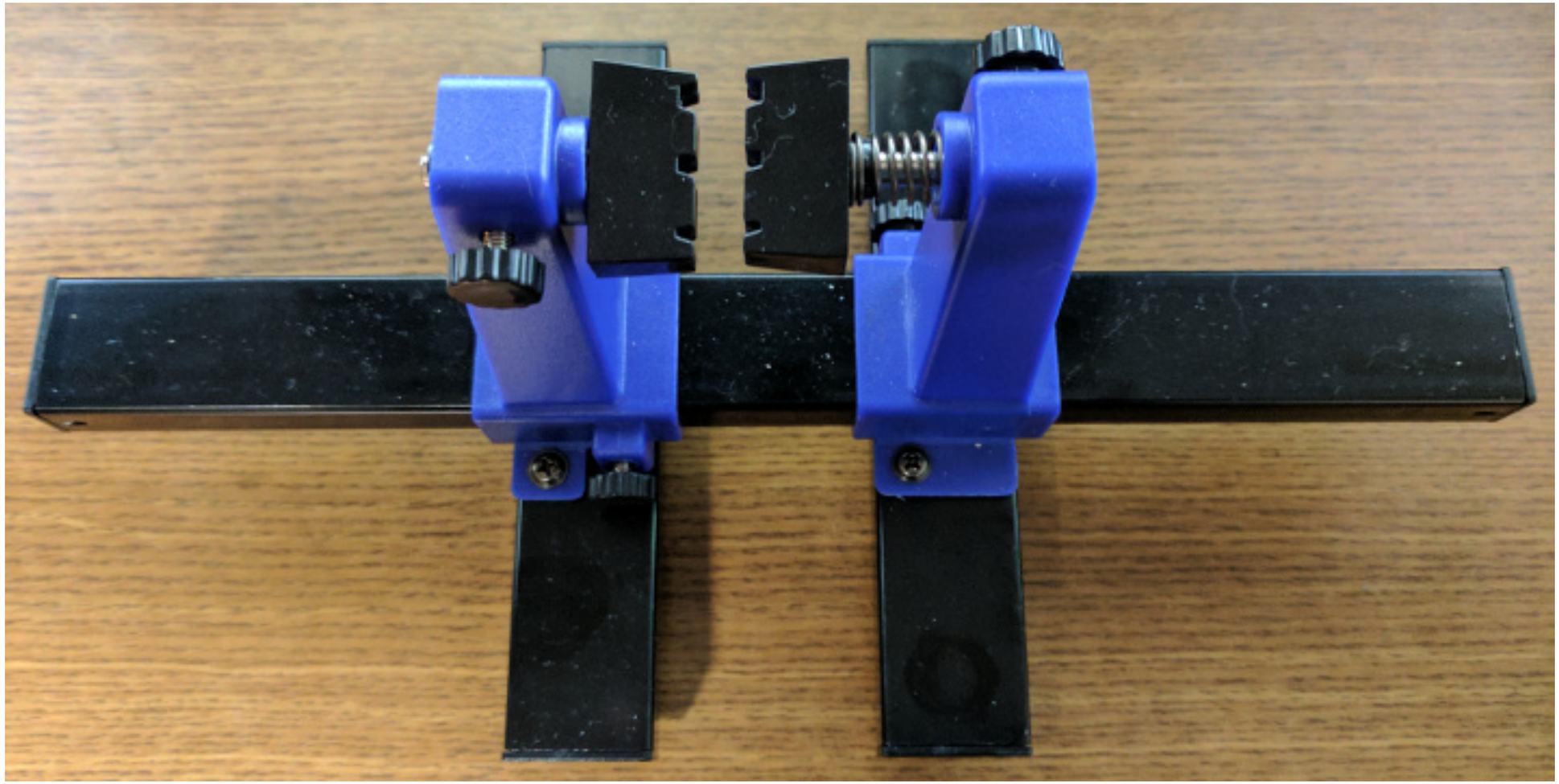

Tweet-a-Watt
KILL A WATT®
METER CLOCK
120.3
Volt
Amp
Watt
Hz
KWH
VA
PF
Hour
python™
Raspberry Pi
cx_Oracle
ORACLE
REST (POST)
REST (GET)
Oracle REST Data Services
ORACLE
JET

A little bit of soldering

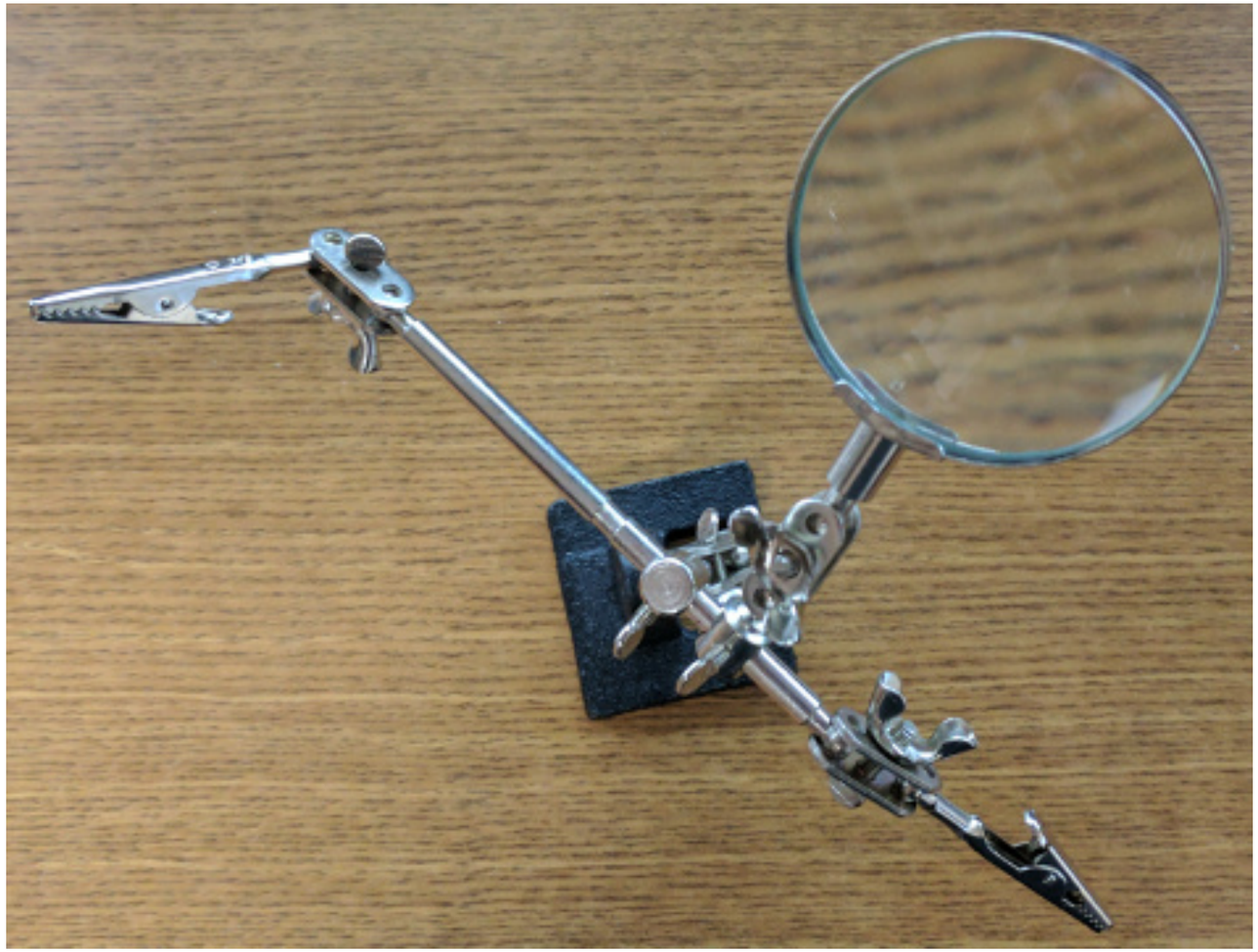


https://learn.adafruit.com/xbee-radios/solder-it

Board Clamp

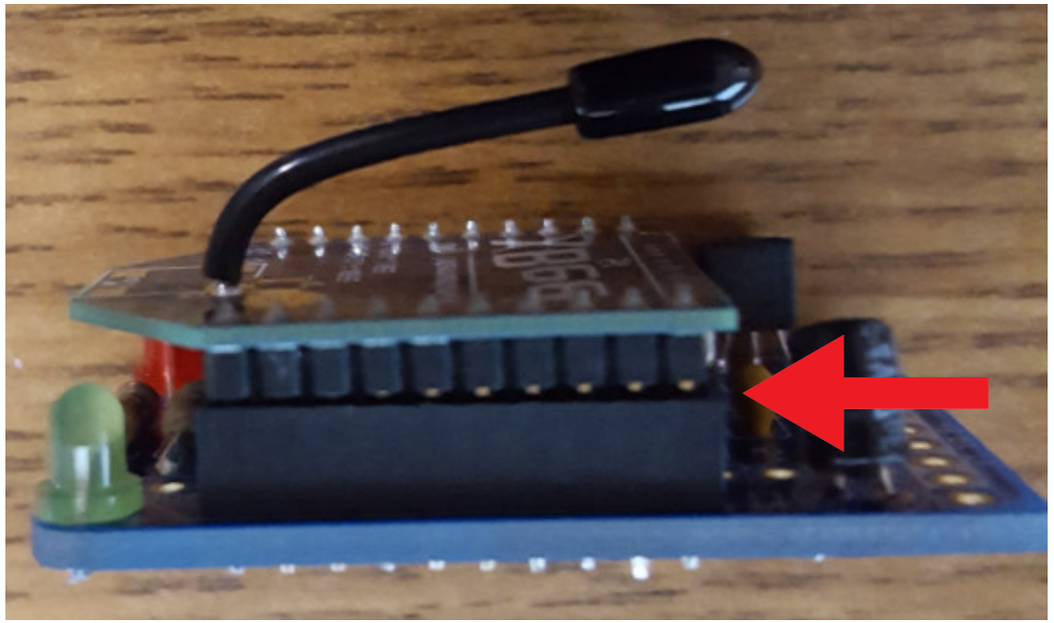


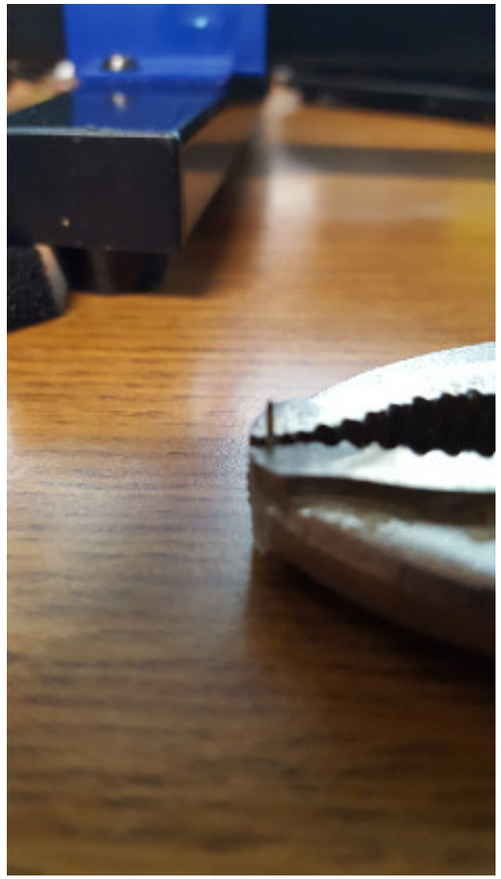
Helping Hands



Receiver Board

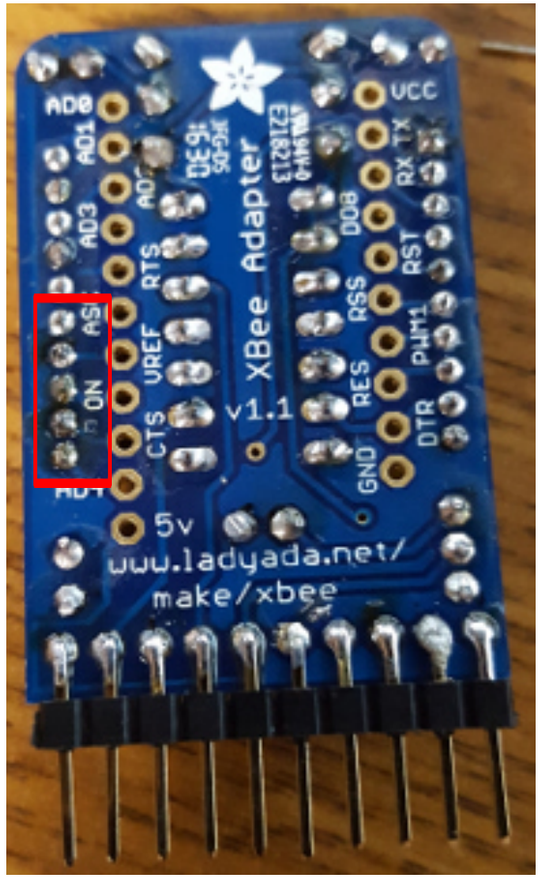
Headers

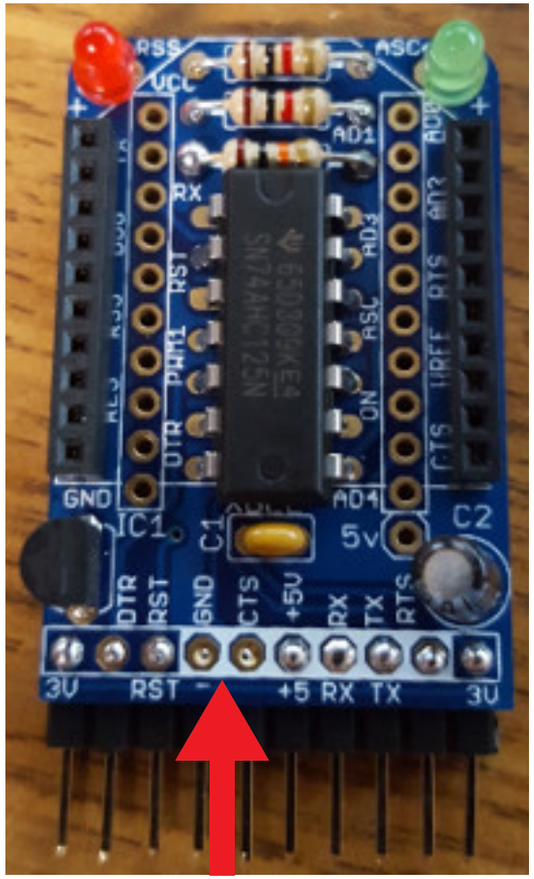


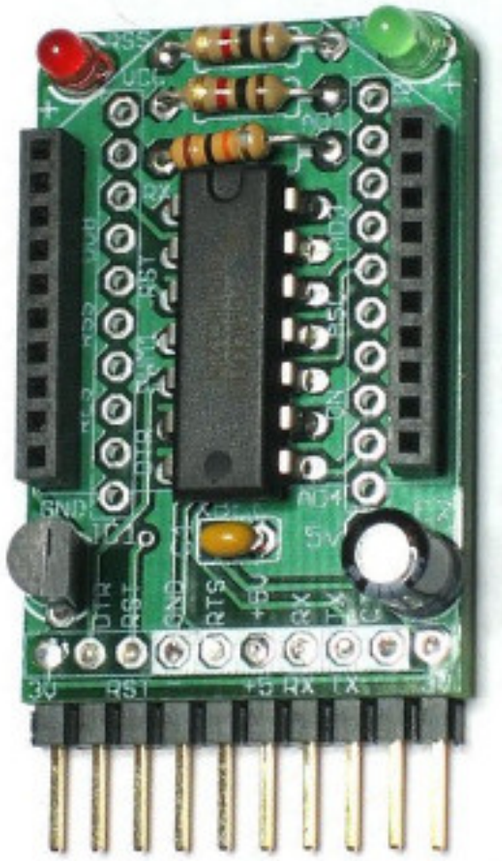


Receiver Board

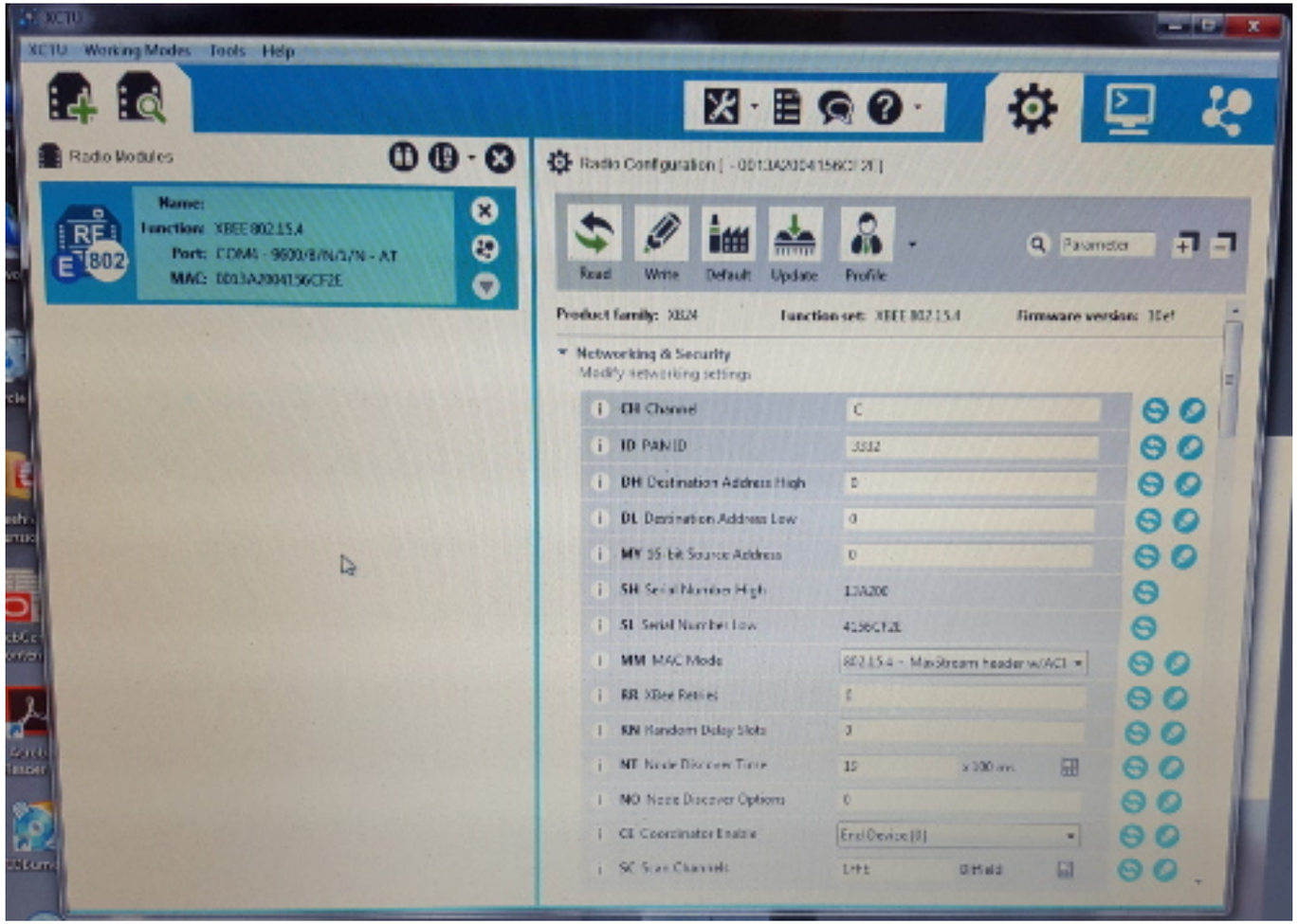
Bottom Header Strip



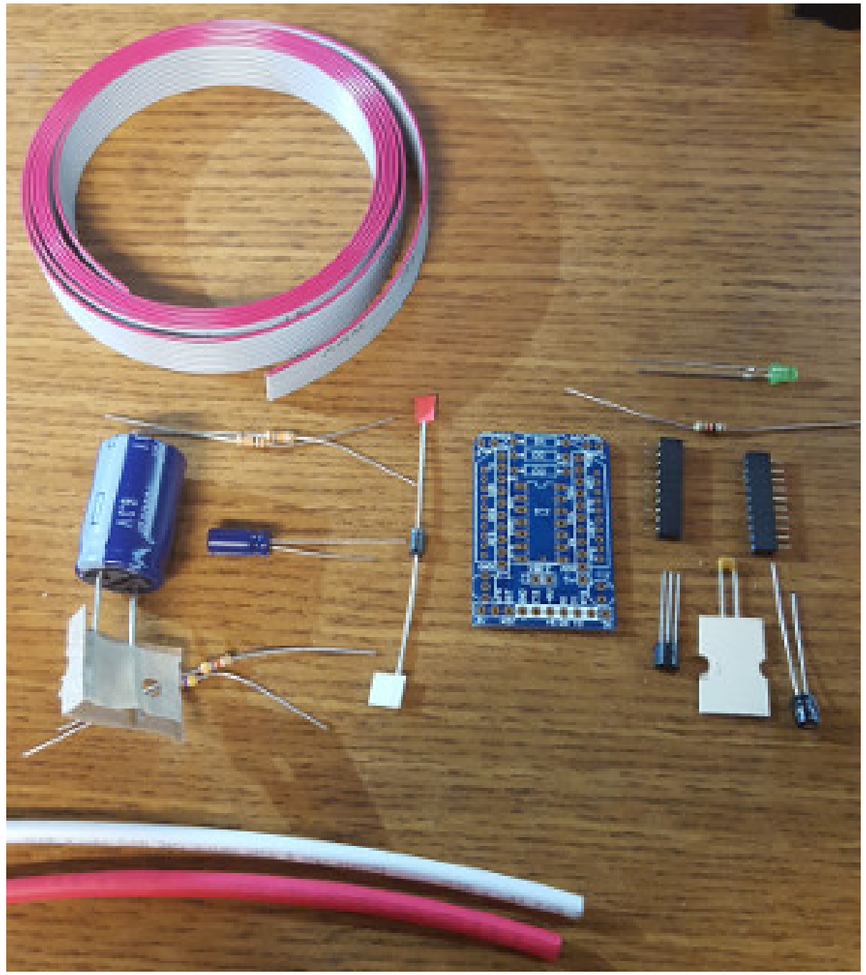




X-CTU

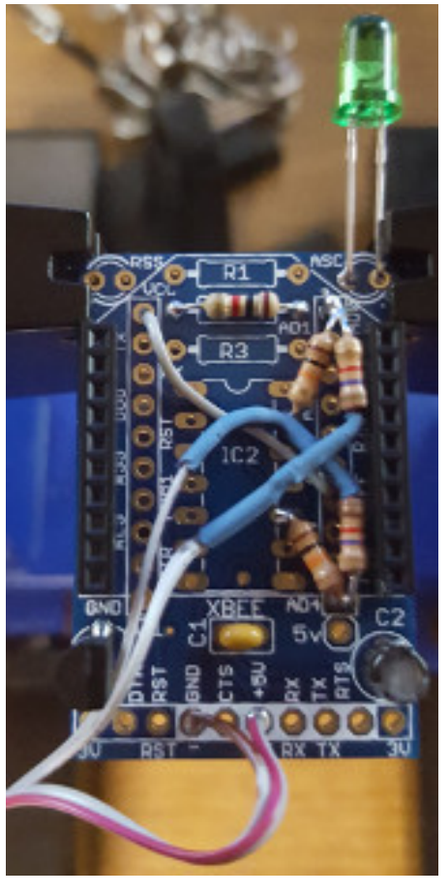

XCTU
XCTU Working Modes Tools Help
Radio Modules
Name:
Function: XBEE 802.15.4
Port: COM4 - 9600/8/N/1/N - AT
MAC: 0013A20041596CF2E
Radio Configuration [- 0013A20041596CF2E]
Read
Write
Default
Update
Profile
Parameter
Product family: XB24
Function set: XBEE 802.15.4
Firmware version: 10e1
Networking & Security
Modify networking settings
CH Channel
ID PAN ID
DH Destination Address High
DL Destination Address Low
MY 16-bit Source Address
SH Serial Number High
SL Serial Number Low
MM MAC Mode
RR XBee Retries
RN Random Delay Slots
NT Node Discover Time
NO Node Discover Options
CE Coordinator Enable
SC Scan Channels

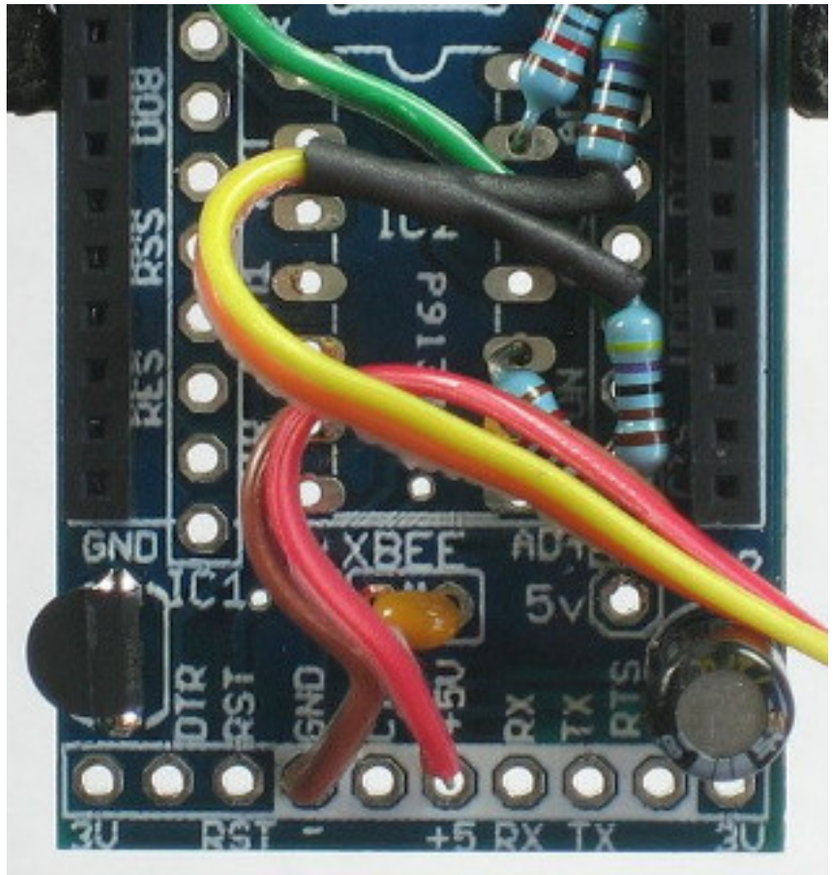
Transmitter



Transmitter

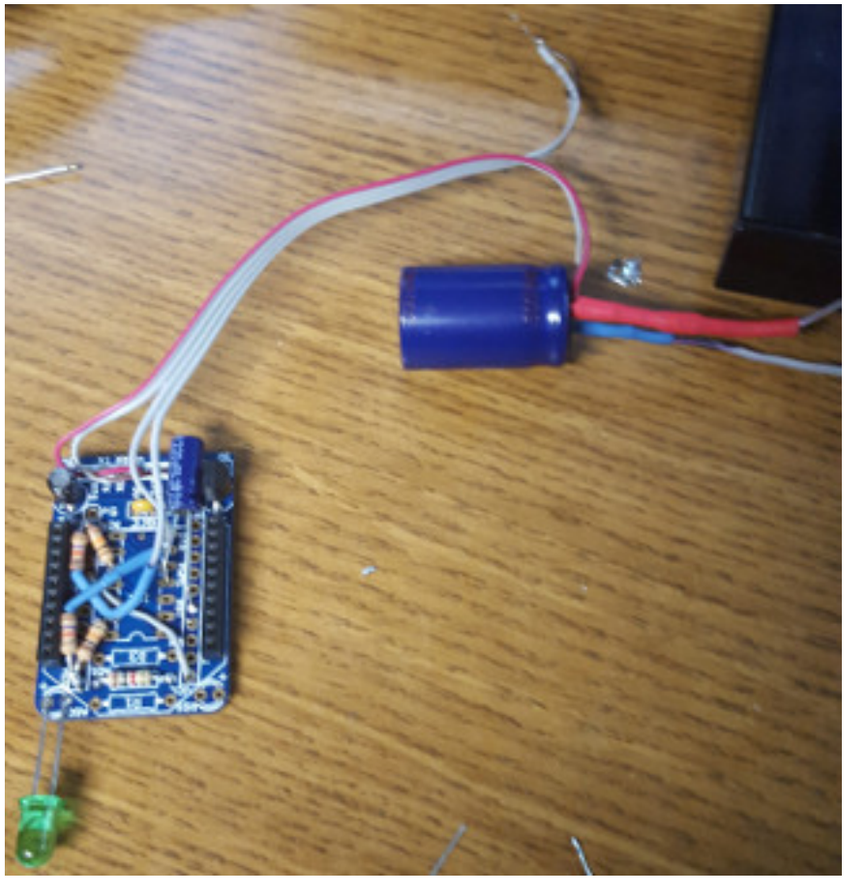
Not Using the Buffer Chip

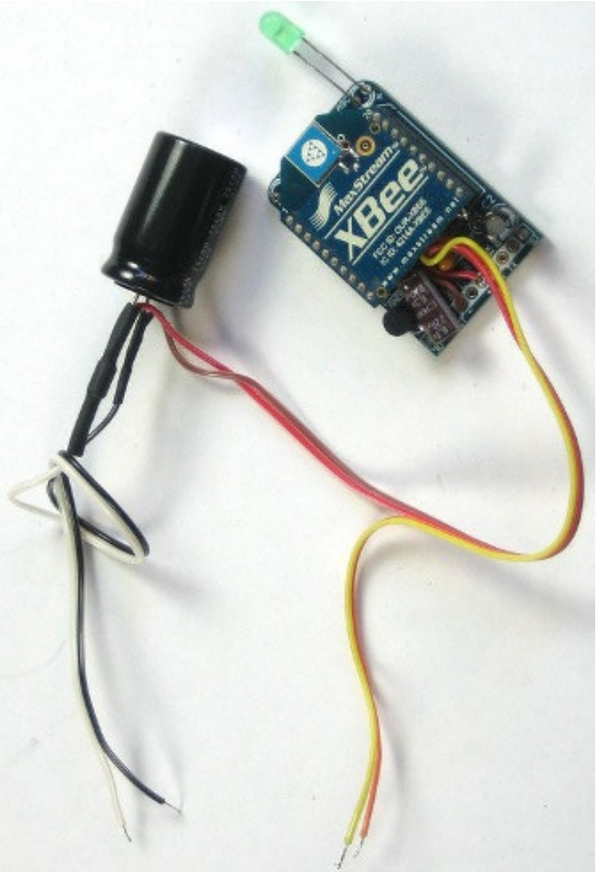




Transmitter

Complete

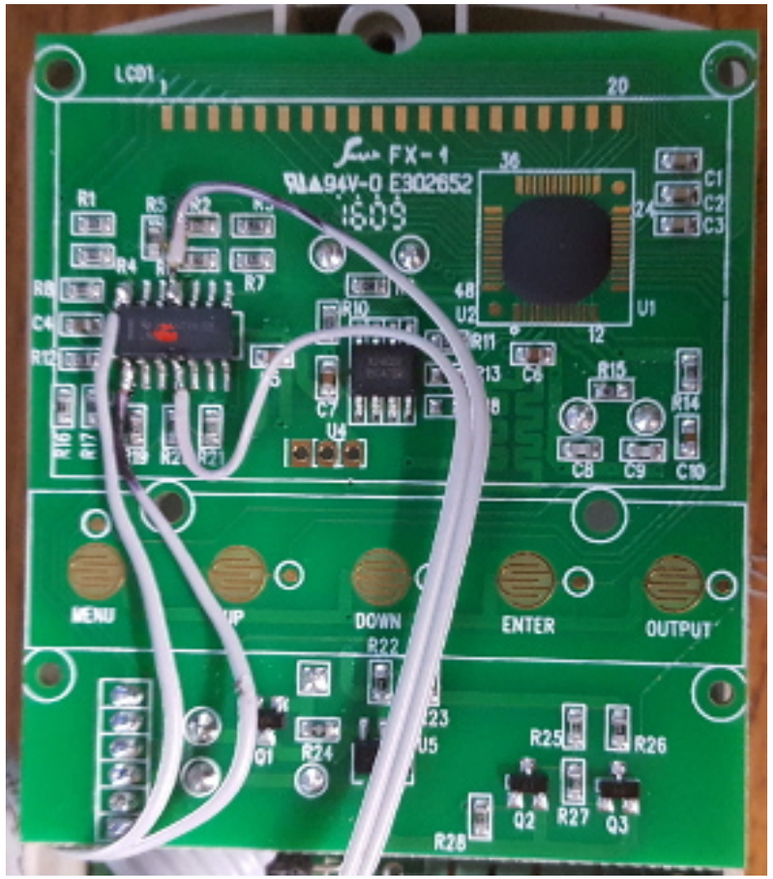


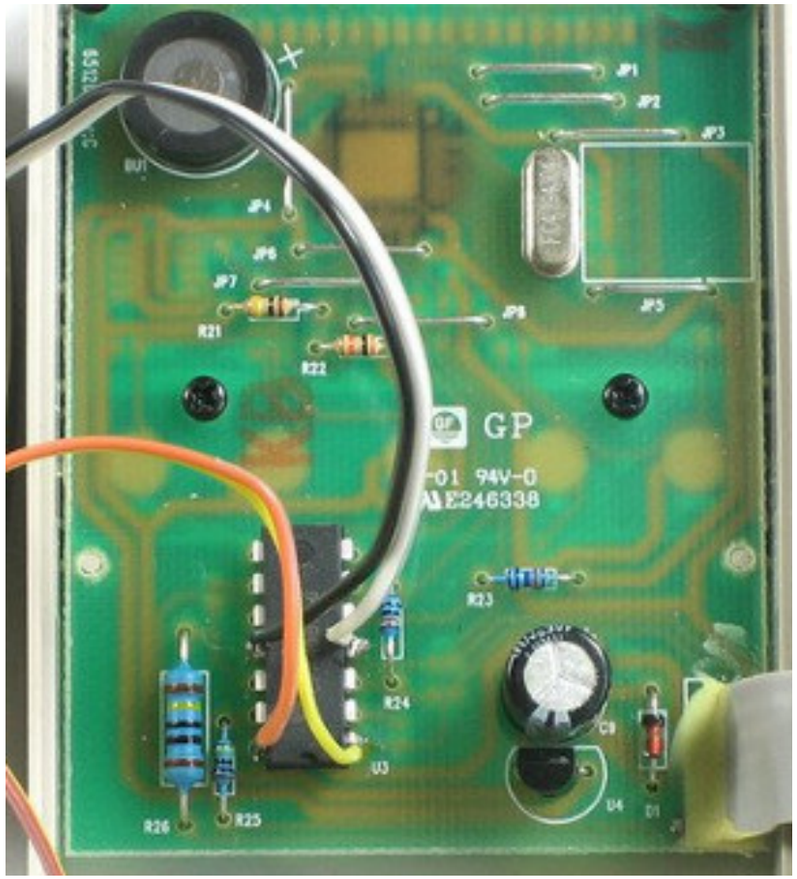


Oh, Crap

Kill-a-Watt

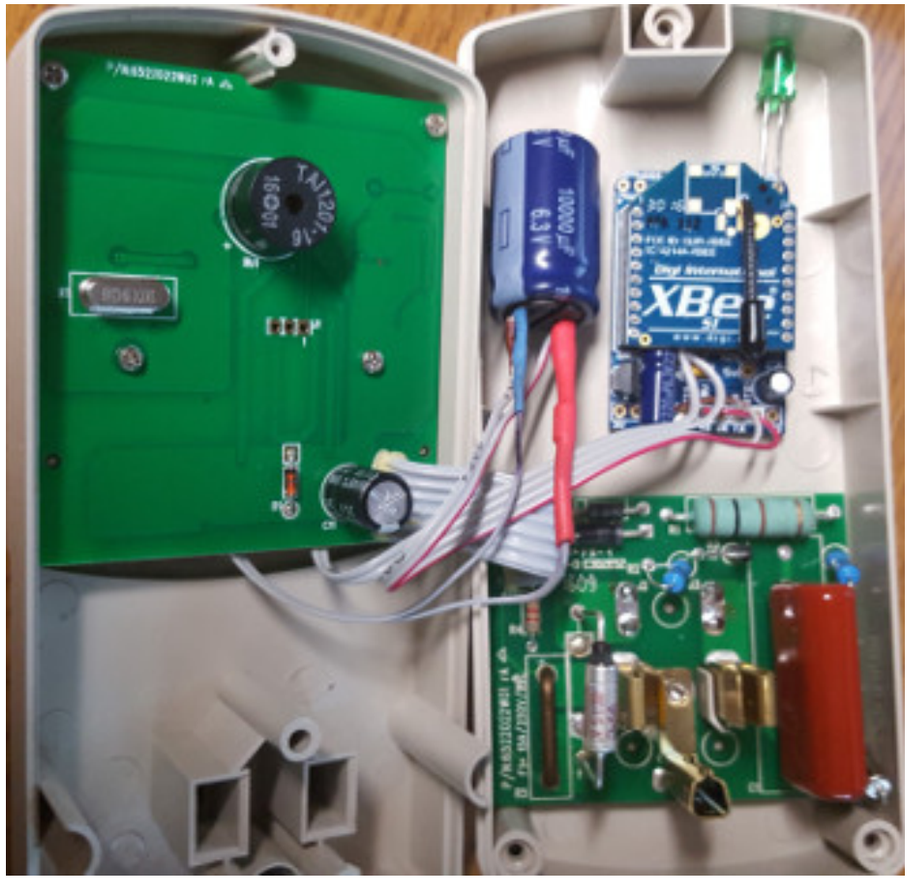
Piggy-back onto the LM2902N chip





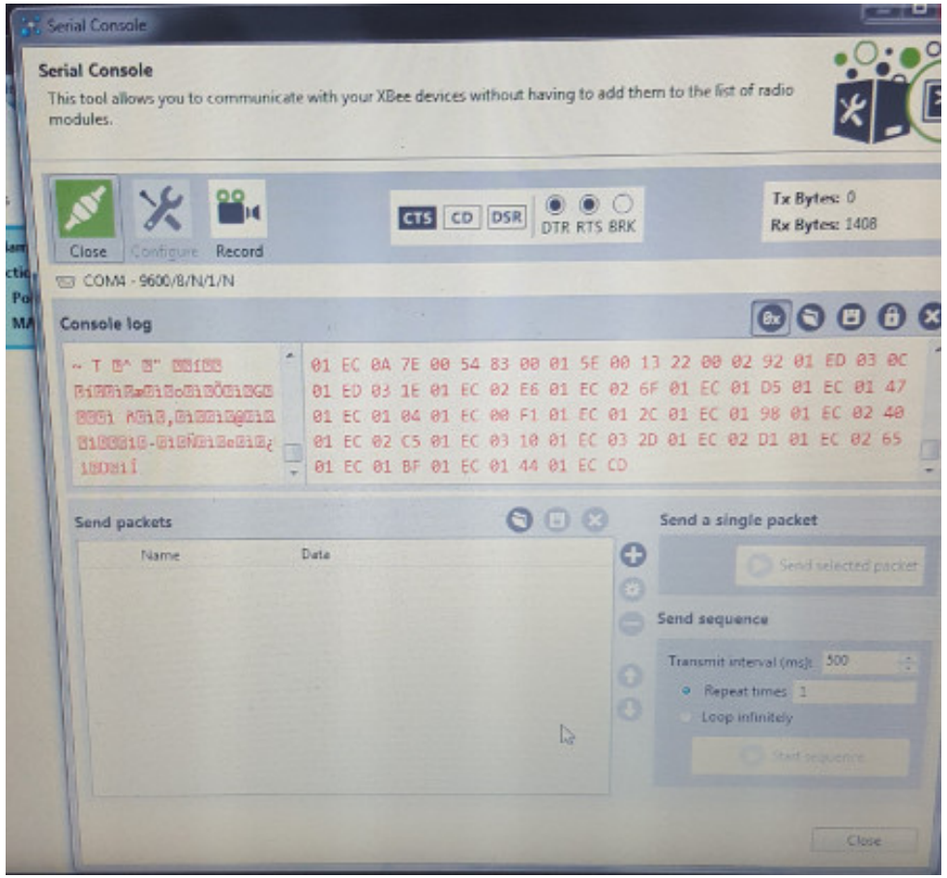
Kill-a-Watt

Position our components

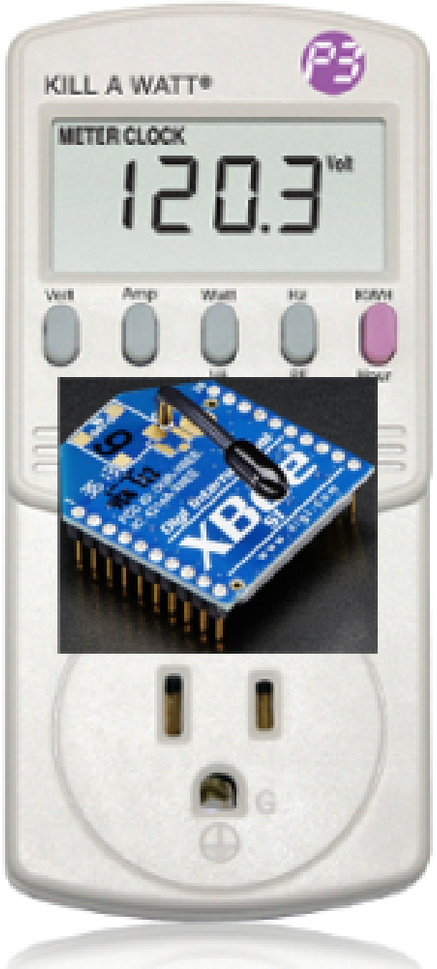


X-CTU

Test the connection



Hardware tips



- Read the guides all the way through before you start.
- Don't clip the component wires before you solder them.
- Good tools are worth the money.
- If you don't try it you won't get better at it.

Persist the data

Welcome to Oracle REST Data Services

http://www.oracle.com/technetwork/developer-tools/rest-data-services/overview/index.html

REST API

- Enable Rest Services for a schema
- Select or create a table
- Create an ORDS module
- Add a Resource Template URI
- Define Resource Handlers for the methods:
  - GET
  - POST

Modify Tweet-a-Watt Code

Tweet-a-Watt-master.zip

python™

xbee.py

- Collect data
- Format data

sensorhistory.py

- Timers
- Sensor data
- Calculates kw/hr
- “5min” just label

wattcher.py

- CSV file
- Twitter interface
- Graph the data
- Google App Engine
- Multiple Kill-a-Watts

wattcher.py -> wattcher-min.py

```
10 SERIALPORT = os.getenv("tawSerialPort") or "COM3"
11 BAUDRATE = os.getenv("tawBaudRate") or 9600

151 print("\t\tWh used in last {} seconds: {}".format(elapsedseconds,dwatthr))

158 if (((time.time() - sensorhistory.fiveminutetimer) >= 10.0) #was 60.0
159  # and (currminute % 5 == 0)

164 avgwattsused = sensorhistory.avgwattover5min()
165 print(time.strftime("%Y %m %d, %H:%M")+", "+str(sensorhistory.sensornum)+",
         "+str(avgwattsused)+"\n")
175       str(avgwattsused)+"\n")
```

Added ords.py

```
import requests
def send_data(sensorId, watt):
  ords_uri = "URLToOrdsServer/kaw_demo/power/usage"
  data = {"sensorId": sensorId,"watt": watt}

  try:
    r = requests.post(ords_uri, data = data, timeout=2.000)
    print("Created, new id: {}\n".format(r.json()['newId']))
  except requests.exceptions.Timeout:
    print("Response is taking too long, moving on\n")
```

Added OracleDb.py

```
import cx_Oracle
pool = cx_Oracle.SessionPool("user", "pw", "dns", min=1, max=10, increment=1)

def insert_data(sensorId, watt):
  conn = pool.acquire()
  cursor = conn.cursor()

  statement =
   'insert into power_usage(sensor_id, watt) values (:1, :2) returning id into :3'

  returned_id = cursor.var(int)
  cursor.execute(statement, (sensorId, watt, returned_id))
  new_id = returned_id.getvalue()[0]

  print("Inserted Record, new id: {:d}\n".format(new_id))
```

ExecuteMany

```
rows = [(2, 1.2),
        (4, 3)]

new_ids = cursor.var(int, arraysize=len(rows))
cursor.setinputsizes(None, None, new_ids)

statement =
 'insert into power_usage(sensor_id, watt) values (:1, :2) returning id into :new_ids'

cursor.executemany(statement, rows)

print("Inserted Records, new ids: {:d}\n".format(new_ids.getvalue()))
```

Call ORDS or cx_Oracle Modules from wattcher-min.py

```
40 ORDS = False

45 if sys.argv[1] == "-o":
46   ORDS = True

167 if ORDS:       # POST data to ORDS
168   ords.send_data(sensorhistory.sensornum, avgwattsused)
or
168   OracleDb.insert_data(sensorhistory.sensornum, avgwattsused)
```

Let's make it pretty

http://www.oracle.com/webfolder/technetwork/jet/index.html

Lots of shiny things

- Voided a warranty / Assembled the hardware
- Created a Rest service with Oracle REST Data Services
- Modified the Python code to connect with cx_Oracle and consume the new ORDS service
- Created a graph with Oracle JET

Try something new

- Hardware / Soldering
- Another programming language
- Build a REST API
- Try a new front end framework

Howdy, I'm Blaine Carter

Oracle Corporation
Developer Advocate for Open Source

Email: blaine.carter@oracle.com
Blog: learncodeshare.net
Twitter: @OraBlaineOS
YouTube: www.youtube.com/blainecarter
Team: community.oracle.com/docs/DOC-917690

ORACLE®