

Data Handling

- Collecting data doesn't mean much if it stays with the sensors.
- Now that you know how to collect sensor data and establish WiFi connections, we'll learn how to move data and store it.
- The first method will be sending a UDP packet with the sensor data in a text format. Each sensor data will be separated by a comma.

- Enter the program into Thonny. Save it as **xmit1.py**
- The program to the right uses the **connect()** function to set up the WiFi network and wait for the laptop to send a command to start.
- Once started, the sensors are configured.
- In the loop, the sensor measurements are taken.
- The variable **telem** becomes a string and has the three measurements added with a comma separating them. This is useful for importing into other programs such as a spreadsheet.
- Time is included in the telem. Time is useful to know when the measurements were made.
- The program loops and sends a measurement two times a second.

```
from machine import Pin,I2C
import connect
from bmp280 import BMP280
import ahtx0
import time

s,addr = connect.connect()
bus0 = I2C(0,sda=Pin(4),scl=Pin(5))
bus1 = I2C(1,sda=Pin(2),scl=Pin(3))
bmp = BMP280(bus0)
humi = ahtx0.AHT10(bus1)

while 1:
    press = bmp.pressure
    temp = bmp.temperature
    hum = humi.relative_humidity
    telem = "%d,%0.2f,%0.2f,%0.2f" % \
        (time.time(),press,temp,hum)
    s.sendto(telem.encode(),addr)
    time.sleep(0.5)
```

Thonny Program

- The highlighted line formats the text.
- The text in between the quotes format the text.
- %0.2f is used to format a floating point number. It only puts the value in the text with two decimal points. The comma becomes part of the text.
 - Below is an example
 - 93432,100901.21,26.21,47.32

```
from machine import Pin,I2C
import connect
from bmp280 import BMP280
import ahtx0
import time

s,addr = connect.connect()
bus0 = I2C(0,sda=Pin(4),scl=Pin(5))
bus1 = I2C(1,sda=Pin(2),scl=Pin(3))
bmp = BMP280(bus0)
humi = ahtx0.AHT10(bus1)

while 1:
    press = bmp.pressure
    temp = bmp.temperature
    hum = humi.relative_humidity
    telem = "%d,%0.2f,%0.2f,%0.2f" % \
        (time.time(),press,temp,hum)
    s.sendto(telem.encode(),addr)
    time.sleep(1)
```

Thonny Program

Receiving Telemetry

- The program entered into Thonny is the transmitter.
- Now, the receiver program will reside on the laptop.
- Open Python IDLE and create a new program.
- This program is similar to the **trigger.py** program from the WiFi lesson.
- The only difference is the **decode()** function which converts binary text to a string.
- Save the name as **receiver.py**.
- Run the xmit program in Thonny on the SLATE board.
- Connect the laptop to the SLATE board.
- Run the receiver program.
- You should see the measurements appearing twice a second.

```
import socket

addr = (('192.168.4.1',10000))
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind(('192.168.4.16',10000))
s.sendto(b'start',addr)

while 1:
    d,a = s.recvfrom(256)
    d = d.decode()
    print(d)
```

Laptop Program

- Data can be saved to a file.
- The CSV format will be used as it is an industry standard.
- CSV is a simple format that uses commas to separate data values and separate lines for each set of data.
- You can load a CSV file into a spreadsheet.
- This requires the same type data values to remain in the same columns.
- Update the **receiver.py** program as shown and save it as **receiver1.py**.

```
import csv
import socket

addr = (('192.168.4.1', 10000))
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('192.168.4.16', 10000))
s.sendto(b'start', addr)

header = ['Time', 'Pressure', 'Temperature', 'Humidity']
fd = open('output.csv', 'w', newline='')
dat = csv.writer(fd)
dat.writerow(header)
try:
    while 1:
        d, a = s.recvfrom(256)
        d = d.decode()
        print(d)
        d = d.split(',')
        dat.writerow(d)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

- First, the csv module is imported to the program
- After the networking section of the code, a header is created. This is the first line in the CSV that identifies the content of each column.
 - It is created as a list. Each element in the list represents a column.
- Next, a file is opened named **output.csv**. The name of the file is set to output.csv. The argument 'w' specifies the file will be written. The newline="" argument is used to keep blank lines from being added to the file.
- dat becomes a csv write object.
- writerow() writes a line of data to the csv file. The data needs to be in a list.

```
import csv
import socket

addr = (('192.168.4.1',10000))
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind(('192.168.4.16',10000))
s.sendto(b'start',addr)

header = ['Time','Pressure','Temperature','Humidity']
fd = open('output.csv','w',newline='')
dat = csv.writer(fd)
dat.writerow(header)
try:
    while 1:
        d,a = s.recvfrom(256)
        d = d.decode()
        print(d)
        d = d.split(',')
        dat.writerow(d)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

- This next section is changed.
- **try** is a keyword in handling exceptions. It's a way to catch an error in program and do something about it.
- **try** is used to detect the closing the program.
- When you try to stop the program, an exception is generated. Specifically the **KeyboardInterrupt** exception.
- The code after **except** is executed which closes the CSV file when the program is stopped.

```
import csv
import socket

addr = (('192.168.4.1', 10000))
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('192.168.4.16', 10000))
s.sendto(b'start', addr)

header = ['Time', 'Pressure', 'Temperature', 'Humidity']
fd = open('output.csv', 'w', newline='')
dat = csv.writer(fd)
dat.writerow(header)
try:
    while 1:
        d, a = s.recvfrom(256)
        d = d.decode()
        print(d)
        d = d.split(',')
        dat.writerow(d)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

- After the data received is decoded into a string type variable, it is split into a list and saved to the same variable **d**.
- The data is saved as a row of data in the CSV file.

```
import csv
import socket

addr = (('192.168.4.1',10000))
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind(('192.168.4.16',10000))
s.sendto(b'start',addr)

header = ['Time','Pressure','Temperature','Humidity']
fd = open('output.csv','w',newline='')
dat = csv.writer(fd)
dat.writerow(header)
try:
    while 1:
        d,a = s.recvfrom(256)
        d = d.decode()
        print(d)
        d = d.split(',')
        dat.writerow(d)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

- Run the xmit program on the SLATE.
- Connect the laptop to the SLATE.
- Run the receiver program in IDLE.
- Collect a few minutes of data.
- In the IDLE window that is displaying the data, press Ctrl and the C key at the same time. This should stop the program.

```
import csv
import socket

addr = (('192.168.4.1',10000))
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('192.168.4.16',10000))
s.sendto(b'start',addr)

header = ['Time','Pressure','Temperature','Humidity']
fd = open('output.csv','w',newline='')
dat = csv.writer(fd)
dat.writerow(header)
try:
    while 1:
        d,a = s.recvfrom(256)
        d = d.decode()
        print(d)
        d = d.split(',')
        dat.writerow(d)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

Using CSV File

- Start a spreadsheet program such as Excel.
- Open the file **output.csv**. It will be located in the same location as your python program on the laptop.
- Make sure the comma is selected as the separator to delimiter. This is used to separate the data into separate columns.
- The data should be listed in four columns.

Time	Pressure	Temperature	Humidity
1783845202	101100.67	22.95	71.54
1783845203	101103.57	22.95	71.53
1783845204	101101.05	22.95	71.54
1783845205	101101.44	22.96	71.56
1783845206	101107.62	22.96	71.59
1783845207	101101.83	22.96	71.56
1783845209	101101.44	22.96	71.56
1783845210	101104.72	22.96	71.56
1783845211	101103.66	22.96	71.55
1783845212	101100	22.96	71.57
1783845213	101102.89	22.96	71.6
1783845214	101104.34	22.96	71.61
1783845216	101100.38	22.96	71.66
1783845217	101105.8	22.96	71.63

- Data can be organized for easier handling.
- We'll use the dictionary which is a data structure that saves data as key-value pairs. Each value will be associated with a unique name.
- Dictionaries work well with JSON files which are files that also contain key-value pairs.
- We'll use the dictionary to hold data. The nice part about dictionaries is each sensor data value is assigned a name. The order of the data is taken care of in the dictionary.
- We'll use the JSON module to move the data.
- Create a new program and call it **xmit2.py**.
- The dictionary created is called **telem**.
- the **dumps()** function converts the telem dictionary into a text array.

```
import time
from json import dumps
from machine import Pin, I2C
from bmp280 import BMP280
import ahtx0
import connect

s, addr = connect.connect()
bus0 = I2C(0, sda=Pin(4), scl=Pin(5))
bus1 = I2C(1, sda=Pin(2), scl=Pin(3))
bmp = BMP280(bus0)
humi = ahtx0.AHT10(bus1)

telem = {}

while 1:
    telem['temp'] = humi.temperature
    telem['humi'] = humi.relative_humidity
    telem['press'] = bmp.pressure
    telem['time'] = time.time()
    print(telem)
    dat = dumps(telem)
    s.sendto(dat, addr)
    time.sleep(1)
```

- Data can be organized for easier handling.
- We'll use the dictionary which is a data structure that saves data as key-value pairs. Each value will be associated with a unique name.
- Dictionaries work well with JSON files which are files that also contain key-value pairs.
- We'll use the dictionary to hold data. The nice part about dictionaries is each sensor data value is assigned a name. The order of the data is taken care of in the dictionary.
- We'll use the JSON module to move the data.
- Create a new program and call it **xmit2.py**.
- The dictionary created is called **telem**.
- the JSON **dumps()** function converts the **telem** dictionary into a text array to be sent to the laptop.

```
import time
from json import dumps
from machine import Pin, I2C
from bmp280 import BMP280
import ahtx0
import connect

s, addr = connect.connect()
bus0 = I2C(0, sda=Pin(4), scl=Pin(5))
bus1 = I2C(1, sda=Pin(2), scl=Pin(3))
bmp = BMP280(bus0)
humi = ahtx0.AHT10(bus1)

telem = {}

while 1:
    telem['temp'] = humi.temperature
    telem['humi'] = humi.relative_humidity
    telem['press'] = bmp.pressure
    telem['time'] = time.time()
    print(telem)
    dat = dumps(telem)
    s.sendto(dat, addr)
    time.sleep(1)
```

JSON on the Laptop

- Enter the program in Python IDLE. Save it as **telemrec.py**.
- From the json module, only the **loads** function is loaded.
- After the network connection, the csv file is created and configured for writing.

```
import csv
import socket
from json import loads

addr = (('192.168.4.1',10000))
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind(('192.168.4.16',10000))
s.sendto(b'start',addr)

fd = open('output.csv','w',newline='')
firstrun = 1
try:
    while 1:
        d,a = s.recvfrom(256)
        telem = loads(d)
        print(telem)
        if firstrun == 1:
            dat = csv.DictWriter(fd,fieldnames=telem.keys())
            dat.writeheader()
            firstrun = 0
        dat.writerow(telem)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

Laptop Program

JSON on the Laptop

- Variable **firstrun** is set to value 1. This is needed because we need to write a header after receiving the first data.
- In the while loop, the program waits for data from the SLATE board.
- Once received, the **loads()** function decodes the JSON data stream into a dictionary object.
- If this is the first data received, the CSV Dictionary writer is created and determines the number of columns will be saved in the CSV file.
- The header is then written using the **writeheader()** function. The **writeheader()** function uses the keys as the column names.
- The variable **firstrun** is set to 0 so that the header writing is not executed again.

```
import csv
import socket
from json import loads

addr = (('192.168.4.1',10000))
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind(('192.168.4.16',10000))
s.sendto(b'start',addr)

fd = open('output.csv','w',newline='')
firstrun = 1
try:
    while 1:
        d,a = s.recvfrom(256)
        telem = loads(d)
        print(telem)
        if firstrun == 1:
            dat = csv.DictWriter(fd,fieldnames=telem.keys())
            dat.writeheader()
            firstrun = 0
        dat.writerow(telem)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

Laptop Program

JSON on the Laptop

- The data is then written and the loop repeats until the program is stopped.
- Run the xmit2.py program in Thonny.
- Connect the laptop WiFi to the SLATE board.
- Run this program.
- Let data collect for a while.
- Stop the program.
- Load output.csv into a spreadsheet program.
- Check out the header at the top. It should be the key names from the dictionary.

```
import csv
import socket
from json import loads

addr = (('192.168.4.1',10000))
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('192.168.4.16',10000))
s.sendto(b'start',addr)

fd = open('output.csv','w',newline='')
firstrun = 1
try:
    while 1:
        d,a = s.recvfrom(256)
        telem = loads(d)
        print(telem)
        if firstrun == 1:
            dat = csv.DictWriter(fd,fieldnames=telem.keys())
            dat.writeheader()
            firstrun = 0
            dat.writerow(telem)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

Laptop Program

- There is another method to send data that is an industry standard called CBOR
- CBOR stands for Concise Binary Object Representation
- The standard is a highly efficient binary data serialization format for sending data.
- Data is compressed into a smaller size to reduce storage needs and increase data transfer rates.
- It is widely used in Internet of Things where resources are constrained.
- It is more efficient than sending text version of data and gets better as more data is added.

- From Canvas, download the **cbor** module. It is a folder with a few python files in it.
- Using the File View in **Thonny**, locate the cbor folder.
- Make sure the SLATE board is plugged in and connected to Thonny.
- Find the folder **cbor** and right click on it and select **Upload**.
- The folder should now appear in the SLATE file list.
- Next, open a power shell or terminal.
- Enter the command **pip3 install cbor2**.
 - If pip3 doesn't work, try pip.

CBOR on the SLATE

- Save the xmit2.py program to xmit3.py on the SLATE board in Thonny.
- Change **json** to **cbor.cbor**.
- No other changes are needed.
- Notice the import specifies **cbor.cbor**. This is because the module is in the folder cbor. This is how modules are referenced in folders.
- The CBOR module compresses the dictionary into a bytearray.
- The data is sent the same way as the JSON data.

```
import time
from cbor.cbor import dumps
from machine import Pin, I2C
from bmp280 import BMP280
import ahtx0
import connect

s, addr = connect.connect()
bus0 = I2C(0, sda=Pin(4), scl=Pin(5))
bus1 = I2C(1, sda=Pin(2), scl=Pin(3))
bmp = BMP280(bus0)
humi = ahtx0.AHT10(bus1)

telem = {}

while 1:
    telem['temp'] = humi.temperature
    telem['humi'] = humi.relative_humidity
    telem['press'] = bmp.pressure
    telem['time'] = time.time()
    print(telem)
    dat = dumps(telem)
    s.sendto(dat, addr)
    time.sleep(1)
```

Thonny Program

CBOR on the Laptop

- In the **telemrec.py**, change **json** to **cbor2**.
- From the **cbor2** module, only the **loads** function is loaded.
- The rest of the program stays the same.

```
import csv
import socket
from cbor2 import loads

addr = (('192.168.4.1',10000))
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind(('192.168.4.16',10000))
s.sendto(b'start',addr)

fd = open('output.csv','w',newline='')
firstrun = 1
try:
    while 1:
        d,a = s.recvfrom(256)
        telem = loads(d)
        print(telem)
        if firstrun == 1:
            dat = csv.DictWriter(fd,fieldnames=telem.keys())
            dat.writeheader()
            firstrun = 0
            dat.writerow(telem)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

Laptop Program

CBOR on the Laptop

- The data is then written and the loop repeats until the program is stopped.
- Run the xmit3.py program in Thonny.
- Connect the laptop WiFi to the SLATE board.
- Run this program.
- Let data collect for a while.
- Stop the program.
- Load output.csv into a spreadsheet program.
- Check out the header at the top. It should be the key names from the dictionary.

```
import csv
import socket
from cbor2 import loads

addr = (('192.168.4.1',10000))
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(('192.168.4.16',10000))
s.sendto(b'start',addr)

fd = open('output.csv','w',newline='')
firstrun = 1
try:
    while 1:
        d,a = s.recvfrom(256)
        telem = loads(d)
        print(telem)
        if firstrun == 1:
            dat = csv.DictWriter(fd,fieldnames=telem.keys())
            dat.writeheader()
            firstrun = 0
            dat.writerow(telem)
except KeyboardInterrupt:
    print("Ending Program")
    fd.close()
```

Laptop Program

