2017

Raspberry Pi Sensor Array

Justin J. Taylor
Concordia University - Portland

Shawn Daley
Concordia University - Portland, sdaley@cu-portland.edu

Matthew E. Wise
Concordia University - Portland, mwise@cu-portland.edu

Raspberry Pi Sensor Array

[1]Justin J. Taylor, [2]Dr. Shawn Daley and [1]Dr. Matthew E. Wise

[1]Math and Science Department, Concordia University, Portland, OR
[2] College of Education, Concordia University, Portland, OR

Abstract

The Raspberry Pi is a fully operational, credit card sized computer that costs $40. When a monitor and keyboard are connected, it can perform any task that a desktop computer can accomplish. The Raspberry Pi uses Linux based operating systems which can be easy to use and learn. Integrating the Pi into projects is made easy by connecting specific hardware components and programming them with the Python computer language. Python is widely used by many companies including Google and Reddit. The Pi is faster and more user-friendly than similar boards such as Arduino. Additionally, there are online support forums from Pi customers and support teams. In the summer of 2017, we were able to outfit the Pi to operate an air quality sensor array.

Project Motivation

For the most part, cheap air quality sensors are located on ground-based sites due to size and performance issues. Using new technology such as the Raspberry Pi computer (Figure 1) and small environmental gas sensors, mobile air quality arrays are possible. One way to mobilize an air quality array is to attach it to an unmanned aerial vehicle (UAV) such as the DJI Spreading Wings S1000+ (Figure 2). Flying a UAV with integrated air quality sensors will allow us to record air quality data in locations that were previously difficult to access (e.g. above a coal train or near a smoke stack).

Figure 1. Raspberry Pi 3

Figure 2. DJI Spreading Wings S1000+.

Sensor Array Hardware

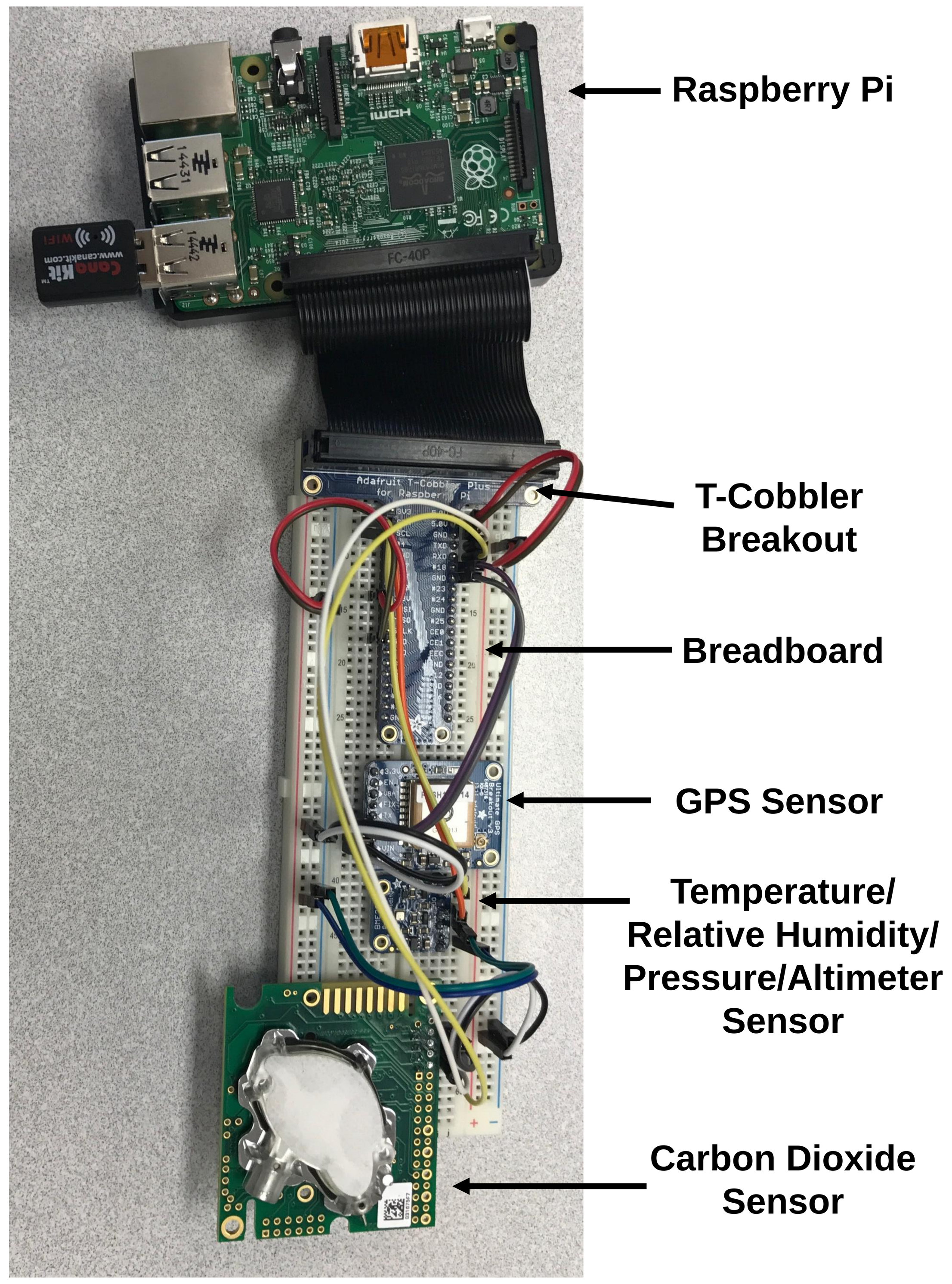

Figure 3. Raspberry Pi and sensor array prototype.

Sensor Array Python Script

```
#Python app to run a K-30 Sensor
import serial
import time
from Adafruit_BME280 import *
import matplotlib.pyplot as plt
import datetime
ser = serial.Serial("/dev/ttyAMA0")
ser.flushInput()
sensor = BME280(p_mode=BME280_OSAMPLE_8, t_mode=BME280_OSAMPLE_2, h_mode=BME280_OSAMPLE_1, filter=BME280_FILTER_16)
print "Serial Connected!"
time.sleep(2)

plt.ion()
x = []
y = []

def graph(co2):
		y.append(co2)
		x.append(time)
		plt.clf()
		plt.scatter(x,y)
		plt.plot(x,y)
		plt.draw()
with open("Test_Sample.csv", "a") as log:
	while True:
			ser.write("\xFE\x44\x00\x08\x02\x9F\x25")
			resp = ser.read(7)
			high = ord(resp[3])
			low = ord(resp[4])
			co2 = (high*256) + low

			#Sets variables from BME280
			celsius = sensor.read_temperature()
			farenheight = celsius * 1.8 + 32
			pascals = sensor.read_pressure()
			hectopascals = pascals / 100
			humidity = sensor.read_humidity()
			# To get pressure in inches of Hg
			inches = pascals * 0.0002953
			# To get Altitude from pressure
			sealevel_pa = 101450.0
			altitude = 44330.0 * (1.0 - pow(pascals / sealevel_pa, (1.0/5.255)))

			print (time.strftime("%H:%M:%S %m/%d/%Y"))
			print ("Temp= {0:} *C".format(celsius))
			print ("Humidity= {0:} %".format(humidity))
			print ("Pressure= {0:} hPa".format(hectopascals))
			print ("Altitude= {0:} meters".format(altitude))
			print ("Co2 = " + str(co2) + "\n")
			log.write(time.strftime("%H:%M:%S %m/%d/%Y \n"))
			log.write("CO2= {0:} PPM \n".format(co2))
			log.write("Temp= {0:} *C \n".format(celsius))
			log.write("Humidity= {0:} % \n".format(humidity))
			log.write("Pressure= {0:} hPa".format(hectopascals))
			log.write("Altitude= {0:} meters".format(altitude))
			log.flush()

			graph(co2)
			time.sleep(.5)
```

Figure 4. Python script used to run sensors.

Eagle Creek Forest Fire

Recently the Columbia Gorge was engulfed in flames by the Eagle Creek forest fire. The fire started on September 2, 2017, and produced a large smoke plume. Eventually, the smoke reached Portland and ash rained downed on the city. We were interested to see changes in ambient carbon dioxide levels as the smoke plume entered Portland and eventually left. Therefore, we placed the Raspberry Pi sensor array onto the roof of a house in NE Portland. The sensor array measured and recorded carbon dioxide levels.

Figure 5. Eagle Creek Forest Fire.

Forest Fire Smoke Data

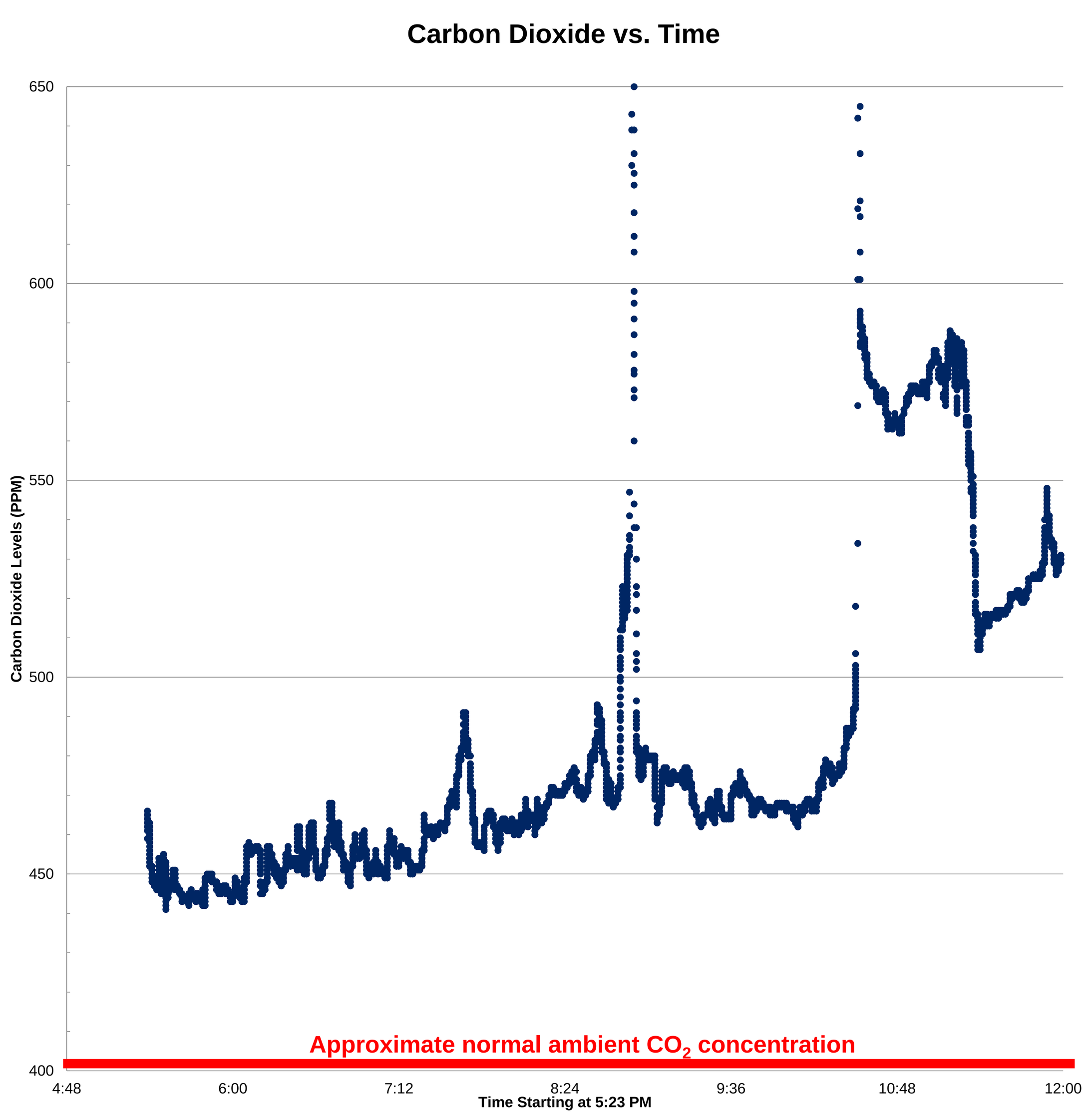

Figure 6. Carbon dioxide concentration (ppm) vs. time

Conclusion

We have not calibrated the carbon dioxide sensor. Therefore, the carbon dioxide concentration data is best used for qualitative purposes. With that being said, there was an increase in carbon dioxide concentrations (above ambient levels of ~400 ppm) as the smoke plume entered Portland. We found the carbon dioxide levels steadily rose as the night came. The rise could be attributed to the lack of photosynthesis in the dark. However, the rise is normally not as drastic as hundreds of parts per million. One carbon dioxide spike occurred at 9 pm (~5000 ppm). We do not think the spike was caused by the smoke plume. It was likely caused by an animal accidentally breathing on the sensor. We noticed the greatest carbon dioxide increases occurred when ash was falling from the sky. We are interested to see how wind speed and direction affected the carbon dioxide levels. Although not shown in Figure 6, the temperature did not affect carbon dioxide levels because temperature remained relatively constant during the night. We wanted to fly the sensor array into the smoke plume and take measurements. However, the UAV is not ready to fly. Hopefully, the system will be ready to take to the skies soon.

Future Work

- Add multiple gas sensors including nitrogen oxide, nitrogen dioxide, carbon monoxide, and ozone.
- Design and 3D print a box that will hold the sensor array on the UAV during flight.
- Conduct additional carbon dioxide sampling in areas that could produce fluctuating levels.

Acknowledgements

- 2017 Concordia University Summer Undergraduate Research Institute
- Murdock College Research Program for the Natural Sciences