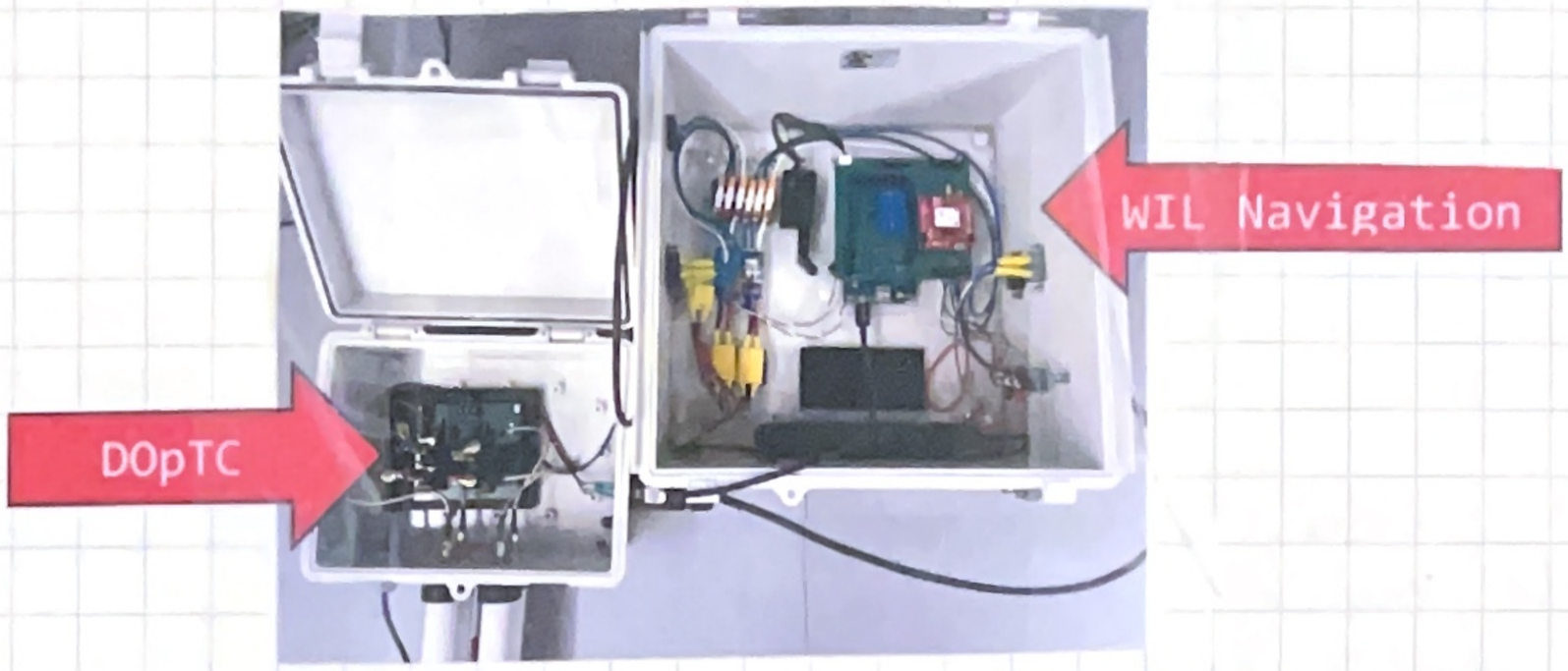
DOpTC Integration With Arduino Giga



Objective

Read the atlas scientific sensors while connected to the I2C bus of the arduino ~~giag~~ giga. The I2C and power is provided through an extension cable.

I2C Scan:

```
#include <Wire.h>

void setup()
{
  Serial.begin(9600);
  while(!Serial) {}
  Wire.begin();
  // I2C Scanner
  for (int address = 0; address < 128; address++)
  {
    Wire.beginTransmission(address); // Sends out the initial address with the write command

    if (Wire.endTransmission() == 0) // Theis means we got an ACK back
    {
      Serial.print("Ox"); Serial.println(address, HEX);
    }
    delay(5);
  }
}
```

COM18

Send

Ox28 → BNO
Ox42 → GPS
Ox61
Ox62
Ox63
Ox64
} Atlas Sensors

DOpTC Integration With Arduino Giga

05/22/25
Wesley Cocke

→ see DOpTC Lab Notebook

The previous code that worked with the MKR 1310 did not work the same on the arduino giga.

The issue came down to not fully clearing the wire buffer between I2C requests.

~~For futu~~ To prevent this issue in future code, make sure you ~~[redacted]~~ do a "Wire.read()" for every byte that you request from a sensor.

Previous code:

requesting 20 bytes
↓

```
Wire.requestFrom(SensorAddress, 20, 1)
while (wire.available())
{
  charIn = Wire.read();
  data[ii] = charIn;
  ii++;
  if (charIn == 0)
  {
    ii=0;
    break;
  }
} Wire.endTransmission();
```

This stops early. If we request 20 bytes but data is only in the first 7 bytes, we will have 13 bytes clogging our buffer.

Think about it like this: We asked the sensor for 20 pieces of mail. If it only needs to send "7", ".", "8", "8", then it will fill the extra 16 bytes with null values. But the above code trys to exit on the null value, leaving at least 15 other null values in the buffer.

This would be like reading the 4 important pieces of mail you get and leaving the other 16 pieces of mail in the mail box!

05/22/25
Wesley Cochu

DO pTC Integration with Arduino Giga

New code:

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

const int DOaddress  = 0x61;
const int ORPaddress = 0x62;
const int pHaddress  = 0x63;
const int Caddress   = 0x64;

byte code       = 0;
byte sensorCode = 0;
byte charIn     = 0;
char sensorData[20];
int timeDelay   = 900;
float data;

float measureEco(int sensorAddress)
{
  Wire.beginTransmission(sensorAddress);
  Wire.write(82);           // (R)ead
  Wire.endTransmission();
  delay(timeDelay);

  Wire.requestFrom(sensorAddress, 20, 1);
  code = Wire.read();

  // 1 is success
  // 2 syntax error
  // 254 not ready
  // 255 no data to send

  // Read the data from the sensor
  for (int ii = 0; ii < 20; ii++)
  {
    charIn = Wire.read();
    sensorData[ii] = charIn;
  }
  Wire.endTransmission();
  // Parse the data
  data = atof(sensorData);

  // Reset Variables
  charIn = 0;
  for (int jj = 0; jj < 20; jj++)
  {
    sensorData[jj] = 0;
  }
  return data;
} //end of measureEco()
```

We could add Some error checking...

~~Wire.Read~~ Wire.read() will be called 20 times.

DOpTC Integration With Arduino Giga

05/22/25
Wesley Cooke

With the refactor of the code, reading all the atlas Scientific sensors is easy as this:

```
50 void setup()
51 {
52   Serial.begin(9600);
53   while (!Serial) {}
54   Wire.begin();
55 }
56
57 void loop()
58 {
59   float DO  = measureEco(DOaddress);
60   float ORP = measureEco(ORPaddress);
61   float pH  = measureEco(pHaddress);
62   float C   = measureEco(Caddress);
63
64   Serial.print(DO);  Serial.print(" ");
65   Serial.print(ORP); Serial.print(" ");
66   Serial.print(pH);  Serial.print(" ");
67   Serial.print(C);   Serial.println();
68 }
```

one function!

different addresses.

Out put of code

```
COM18
0x28
0x42
0x61
0x62
0x63
0x64
0.00 664.10 6.46 107.80
0.00 664.20 6.46 108.30
0.00 664.40 6.46 108.00
0.00 664.60 6.46 107.80
0.00 664.30 6.46 107.80
0.00 664.50 6.46 107.70
0.00 664.60 6.46 107.80
0.00 664.80 6.46 107.60
Autoscroll  Show timestamp
```

I2C SCAN