Dissolved Oxygen Meter Programming Notes

The Dissolved Oxygen Meter needs to connect to the project lab the following signals: 3.3V power, ground, and one analog input (Figure 1). The header is single pins, and three male-to-female jumper wires are supplied.

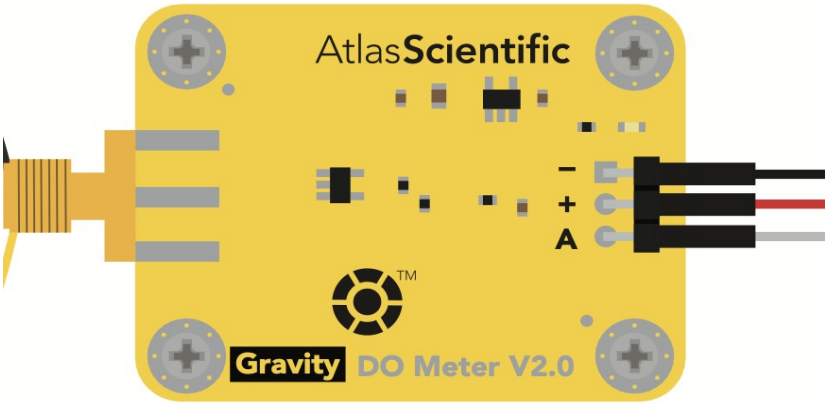

Figure 1: Diagram of Atlas Scientific's Gravity Dissolved Oxygen Meter. Pinout: **-** *= ground,* **+** *= 3.3V power,* ***A*** *= analog voltage output*

To get the needed connections from the project lab, we could use the Grove Analog Connector, which is located on the right side of the project lab (Figure 2). Alternatively, we could use one of the two mikroBUS connectors on the left side, or the screw-terminal connector block at the lower left. Each of these options connects to a different analog input channel, so it is important that the code is able to be initialized with any of the analog input channels. Here, we use the screw-terminals.

Dissolved Oxygen Meter	Screw-Terminal Block on Project Lab
-	GND (pin 3)
+	3.3V (pin 2)
A	A4 (pin 4)

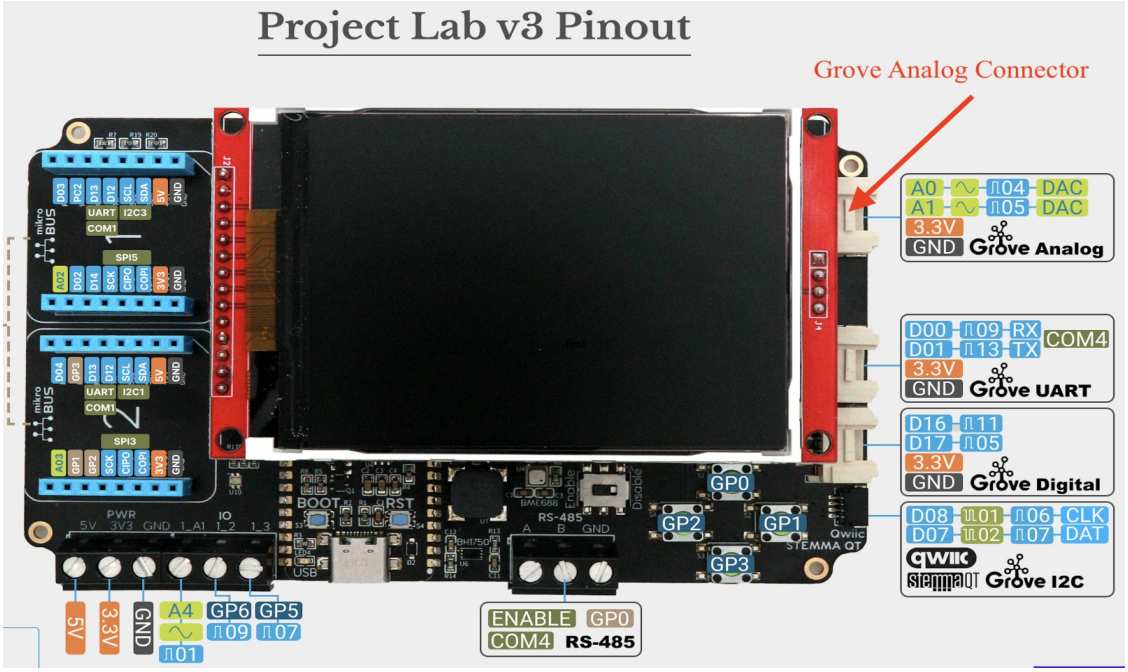

Figure 2: pinout of ProjectLab version 3. Five different channels for analog input are variously available on the Grove analog connector, the mikroBUS connectors, and the screw-terminal IO block.

Development Environment and Hello World

Just to make sure we have the development environment set up properly, we do the classic blink an LED for our first program. Using Visual Studio 2022 community edition on Mac and PC, we setup for Meadow development, downloading and installing the Meadow extension for Visual Studio, and installing the NuGet package for the projectLab board by following the instructions at:

https://developer.wildernesslabs.co/Meadow/Getting_Started/Hello_World/

and

https://www.hackster.io/wilderness-labs/getting-started-with-meadow-s-project-lab-eeb569

Some things to note:
The ProjectLab V.3 has a F7CoreComputeV2 module instead of the F7FeatherV1 or F7FeatherV2, so code samples will need to have the base class for App changed to reflect that.
The ProjectLab with F7CoreComputeV2 does not have an on-board RGB LED, so examples using that LED will not work, because it cannot find the correct pins.
The pinout shown in Figure 2 maps the special function labels silk-screened onto the board to the actual digital IO pin numbers needed in the code, e.g., the mikroBUS 1 Cable Select pin is D14.

The code in Figure 3 will blink at 1Hz an LED connected to D14. Connect the ground rail of a breadboard to GND of the ProjectLab using the screw terminals or mikroBUS headers. Connect

mikroBUS 1 CS pin (D14) to the cathode of an LED, and connect the anode of the LED to ground through an approximatlely 1k resistor. Run the code and watch the LED blink.

```
using Meadow;
using Meadow.Devices;
using Meadow.Foundation.Leds;
using System;
using System.Threading.Tasks;

namespace HelloMeadow
{
    // ProjectLab uses an F7CoreComputeV2 board
    public class MeadowApp : App<F7CoreComputeV2>
    {
        Led led;

        public override Task Run()
        {
            Console.WriteLine("Run...");

            led.StartBlink(TimeSpan.FromSeconds(1), TimeSpan.FromSeconds(0.5));

            return base.Run();
        }

        public async override Task Initialize()
        {
            Console.WriteLine("Initialize...");
            // Change digital pin to D14 and use mikrobus 1 CS pin
            led = new Led(Device.CreateDigitalOutputPort(Device.Pins.D14));

            await base.Initialize();
        }
    }
}
```

Figure 3: Code to blink an LED connected to mikroBUS 1 CS pin

Reading Analog Inputs

The next step is to write code to use the analog input and verify that it is working. Because we want to build a library, we start with one of the MeadowLabs sensor libraries that uses an analog input, in this case a moisture meter:

https://github.com/WildernessLabs/Meadow.Foundation.Grove/tree/main/Source/MoistureSensor

The actual application doesn't matter here because the code in question just returns the analog reading in volts.

Here, we clone the entire repository of projects:

`git clone https://github.com/WildernessLabs/Meadow.Foundation.Grove`

The moisture meter code comes in two projects, one that makes a simple MoistureMeter driver that consists of a single class, and one for a sample app that creates an instance of the class and repeatedly

polls the MoistureMeter object and prints its value to the Meadow console. The following changes were made to the code:

1) The base class for the MoistureSensor_Sample was changed to reflect the ProjectLabV3:

public class MeadowApp : App<F7CoreComputeV2>

2) The device pin used for the sensor in the MoistureSensor_Sample was changed to be A04, available from the screw-terminal block header:

sensor = new MoistureSensor(pin: Device.Pins.A04);

A potentiometer was used for testing the analog input. The high-side and low-side pins of the pot were connected to the ProjectLab 3.3V and GND connections on the screw-terminal block, and the wiper pin was connected to the Analog Input 0 pin on the screw-terminal block. **The analog input pins use a 3.3V reference and can be damaged if connected to voltages above 3.3V.** Upon running the DOsensorDriver application, voltages were continuously read and printed to the console (Figure 4). The values were fairly constant with only a slight jitter, around ±3 mV.

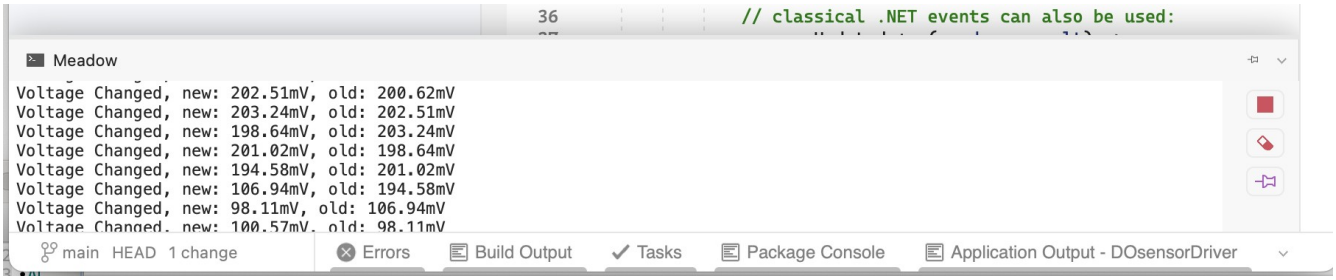

Figure 4: Console output with a potentiometer connected to analog input 0. The first four values shown were with potentiometer steady at 0.2V, after which the pot was turned to 0.1V

The values read by the analog input were compared with values obtained with a multi-meter (Klein Tools MM200). There was good agreement between the two methods (Figure 5), suggesting that the analog input reads values accurately over the entire range from 0 to 3.3V.

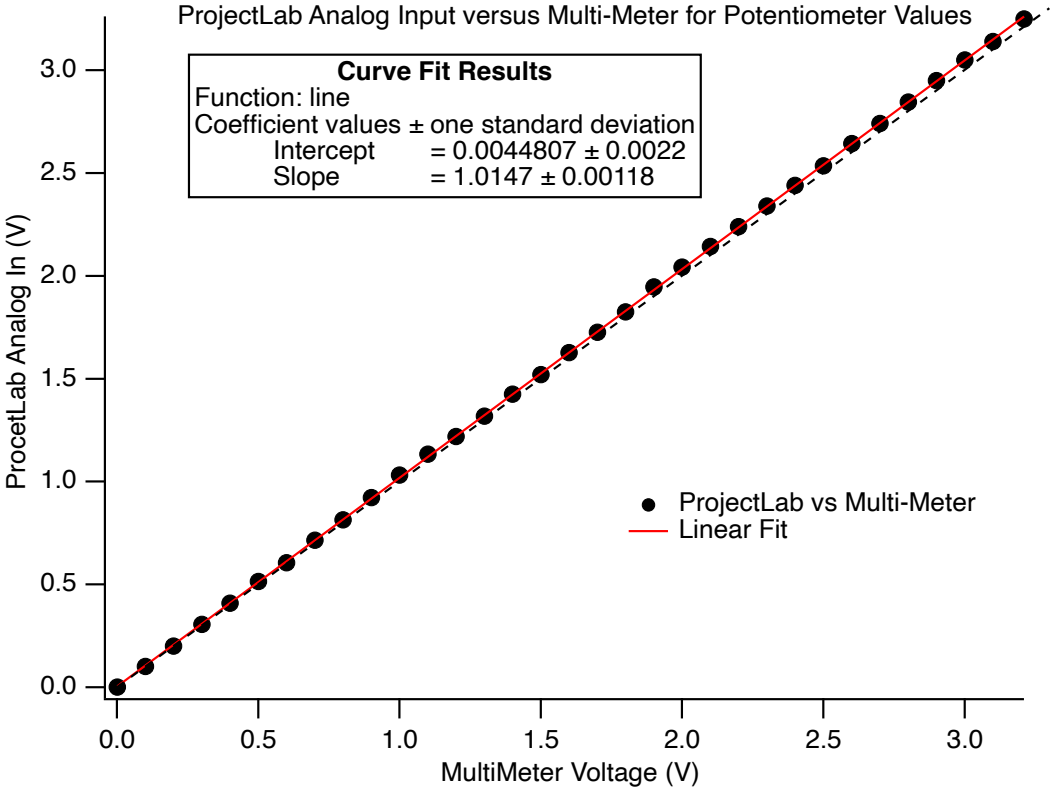

Figure 5: ProjectLab analog input values compared to multi-meter values for voltages generated by a potentiometer. The dashed line for comparison has intercept=0 and slope=1.