TUTORIAL A

**Context Sensing with Phidgets and building the Pressalit Hygiene Sensor**

## Introduction:

Phidgets (<http://www.phidgets.com/>) are used for hardware prototyping pervasive computing prototypes. They belong to a family of hardware prototyping toolkits that include Ardunio, Netdunio, Sun SPOT, FEZ and related technologies.

Phidgets are slightly more sophisticated as compared to most other prototyping toolkits, as they provide extensive programming API’s for most modern OS and programming platforms, including Windows, Linux, iOS, Android, iPhone, as well as MatLab and LabView for research purposes.

The Phidget Interface Kit family of boards allow for easy USB connectivity to a wide range of sensors and actuators, allowing programming of these using your preferred programming environment.

The Phidget SBC family of boards allow for independent operation of devices, by using a Linux based OS. Rather than using USB, it is now also possible to connect using Ethernet or WiFi connections. The Phidget SBC board can be accessed via a native Web service providing a high degree of access transparency, or computation can be done on board, using Java, C# (Mono.Net) or C++.

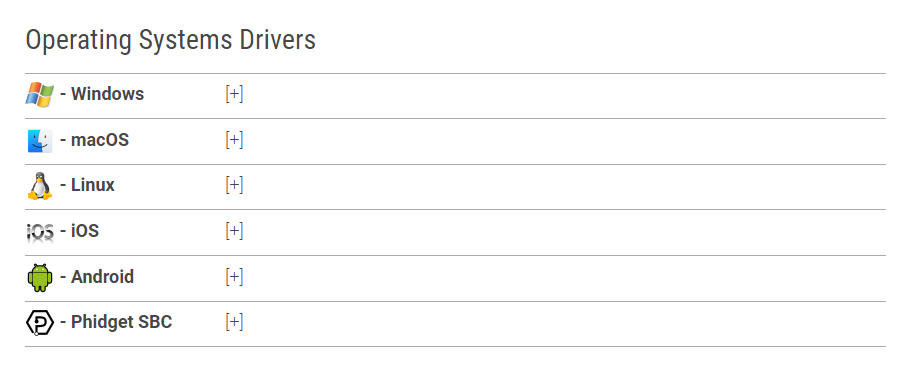
## Phidget Interface Kit:

The PhidgetInterfaceKit 8/8/8 allows you to connect devices to any of 8 analog inputs, 8 digital inputs and 8 digital outputs. It provides a generic, convenient way to interface your PC with various sensor devices. A variance of this is the PhidgetInterfaceKit 2/2/2 which is smaller but cheaper. You shall try to use both types, as well as the SBC kit, and relay switching boards.

The analog inputs are used to measure continuous quantities, such as temperature, humidity, position, pressure, etc. Phidgets offers a wide variety of sensors that can be plugged directly into the board using the cable included with the sensor. These sensors are the same that you may find for Ardunio and other open hardware projects, but Phidgets allow for quite easy access.

**Procedure for the exercises:**

1. Work in your groups!
2. Start with only one kit in the group so that you help each other getting it to work.
3. End up with all of you having tried installing the devices and having coded something relevant
4. Go through the main exercise scenarios (a-f). Focus on at least getting a-c done today.
5. **As there are not enough of all Phidget Interface Kits, at least one group (group 1) should start with exercise scenario b), and then switch with another group when they are done**



**a) Exercises for Phidgets Interface Kit 2/2/2**

1. Start with reading the product manual [**http://www.phidgets.com/docs/1011\_User\_Guide**](http://www.phidgets.com/docs/1011_User_Guide) and download drivers from: <http://www.phidgets.com/docs/Operating_System_Support> (find your OS and follow the instructions). For Phidget22 for Windows, it is: <https://www.phidgets.com/docs/OS_-_Windows#Quick_Downloads> (choose 64 bit for most of your laptop PC’s)
2. Next, install the Phidget Interface Kit drivers and connect to them using the Phidget Control Panel
3. Try out the example applications available from Phidgets (either C#, Java, C++ as you please) <https://www.phidgets.com/?view=code_samples&lang=CSharp>
4. Find an IR reflective sensor, which has a detection range of 4 mm. Try it out and discuss with your group what should a device can be used for?

**b) Exercises for Phidget Interface Kit 8/8/8:**

1. Start with reading the product manual <http://www.phidgets.com/documentation/Archive/1018_1_Product_Manual.pdf> and download drivers from: <http://www.phidgets.com/docs/Operating_System_Support> (find your OS and follow the instructions)
2. Next, install the Phidget Interface Kit drivers and connect to them using the Phidget Control Panel
3. Try out the example applications available from Phidgets (either C#, Java, C++ as you please)
4. Try using the interface kit for accessing the output port by lighting two LED’s in different colors to showcase ambient feedback to a user – by making your own console application
5. Try using the interface kit for accessing the input port by using the pressure sensor mat.
6. Find a Temperature/Humidity sensor, sample the values every 1 second, convert them to celcius and “percentage of humidity” – and display it in a Java, Android, iOS, or Windows C# WPF or Forms program (only the latter two are supported on class).
7. Find a distance sensor or ultrasound sensor and test the precision and accuracy of the sensor. Can you use it to see if a user is moving towards a touch screen computer mounted on the wall?

**c) Exercises with the RFID kit**

The PhidgetRFID Read-Write reads RFID tags that are brought in close proximity to the reader and returns the tag identification number. Writing data to T5577 tags is also supported.

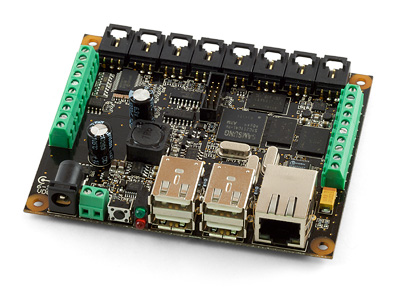
RFID (radio frequency identification) systems use data strings stored inside RFID tags to uniquely identify people or objects when their tags are scanned by an RFID reader. These types of systems are found in many applications such as passport protection, animal identification, inventory control systems, and secure access control systems.

The PhidgetRFID Read-Write supports reading and writing in 3 protocols; EM4100, ISO11785 FDX-B, and PhidgetTag. The PhidgetTag protocol simply stores up to 24 ASCII characters to the tag, eliminating the necessity for a table of corresponding tag numbers and names in your program. Phidgets sells EM4100 read-only tags that can be read with either of our RFID readers, and writable tags which can be written with the 1024 using any protocol. Any 3rd-party EM4100 or ISO11785 tags can be read.

Because passive tags require a strong RF field to operate, their effective range is limited to an area in close proximity to the RFID reader. The distance over which the RFID tag is usable is affected by such things as the tag shape and size, materials being used in the area near the reader, and the orientation of the reader and tag in respect to each other and in their operating environment. The smaller a tag, the closer it must be to the reader to operate.

1. Find an RFID kit and connect it with the phidget panel. Try out holding a tag near to it.
2. Make a C# (or other) console application that will react to seeing an RFID tag placed on it and provide some feedback to the user. It could be a user logging in to a system, or it could be a nurse registering a new device. Discuss where a tag could be placed if it should be used to track a device?
3. Have a look at: <http://www.phidgets.com/products.php?category=14> and discuss what the different types of tags could be used for. Are there any other tags you would like to use?

**d) Exercises with the Phidget SBC**

The PhidgetSBC is a Single Board Computer with an integrated PhidgetInterfaceKit 8/8/8. At its most basic, it can be thought of as a Phidget that you connect using a network cable instead of USB. The PhidgetSBC also provides six full-speed ports that allow you to use normal USB Phidgets over its network connection. This can extend the effective range of a Phidget from USB’s maximum of 15 feet, to anywhere that your network reaches. This is achieved using a web service interface.

The PhidgetSBC exposes an easy to use interface for setting up and running custom applications on-board. This allows the PhidgetSBC to operate autonomously, without the need for a graphical interface or a remote connection at all times.

An integrated PhidgetInterfaceKit 8/8/8 allows you to connect devices to any of 8 analog inputs, 8 digital inputs and 8 digital outputs.

For more advanced users, the PhidgetSBC is an embedded computer that runs Debian GNU/Linux. Full shell access is provided via a built-in SSH server, access to the full Debian package repository. This means you can run Java or C++ applications in the Linux environment. PhidgetSBC2 can also be expanded to run C# and more.

1. Start with reading the guidelines for Phidget SBC installation and configuration: <http://www.phidgets.com/products.php?product_id=1073> (Use [http://www.phidgets.com/documentation/Phidgets/1072.pdf for PhidgetSBC2](http://www.phidgets.com/documentation/Phidgets/1072.pdf%20for%20PhidgetSBC2))
2. Please note – you need a DNS enabled network that will allow the PhidgetSBC to connect and obtain an IP adress. Usually this means that you will need a router unless your computer. Phidget SBC also supports zero configuration networking with Bonjour – but then you will need to have such a device available.
3. Also please note – you may have to do a firmware update on the Phidget SBC to get it running with the current version of the API.
4. Install the PhidgetSBC drivers and connect to it using the Phidget Control Panel
5. Connect the Phidget SBC to your computer with an Ethernet cable
6. Try out the example applications available from Phidgets (either C#, Java, C++ as you please)
7. Try out a range of the available sensors using the Phidgets software API’s.
8. Try attaching the WiFi USB modem including configuring using the Web configuration interface
9. Try attaching the camera and / or audio devices, and access these from a PC application over the internet. See if you can use the camera to monitor a patient. E.g. when a movement sensor notices movement, use the web cam to grab a picture.
10. Try creating a Java or C++ application to upload to the SBC for non-PC interaction. Discuss when you would not want to use the SBC with a PC
11. Try creating an application for your Android phone or tablet
12. Try connecting an RFID board and a Sound Level Board to the PhidgetSBC ports. What happens?
13. Discuss with your group when you would want to use the Phidget Interface kits 2/2/2, 8/8/8 and when to use the PhidgetSBC boards?

**e) Exercises with the Phidgets Interface Kit 0/0/4 (Relays)**

The PhidgetInterfaceKit 0/0/4 provides a convenient way to interface your PC with various higher-voltage devices such as incandescent bulbs, high-power relays, and motors. This includes turning the lights on – or the television for instance. The kit contains 4 Relay Outputs for switching AC or DC power.

1. Start with reading the product manual <http://www.phidgets.com/products.php?category=9&product_id=1014_2> and download drivers from: <http://www.phidgets.com/docs/Operating_System_Support> (find your OS and follow the instructions)
2. Next, install the Phidget Interface Kit drivers and connect to them using the Phidget Control Panel
3. Try turning the lights on and off
4. Build an application where you turn the lights on when there is no longer pressure on the pressure pad. This could be used in the bed of a user, or when someone steps on a doormat or similar.
5. Consider how you may use this to turn the television on when a user is standing in front of it ready to do exercises.

**f) Exercises with the PhidgetBridge 4-Input (Load Cells)**

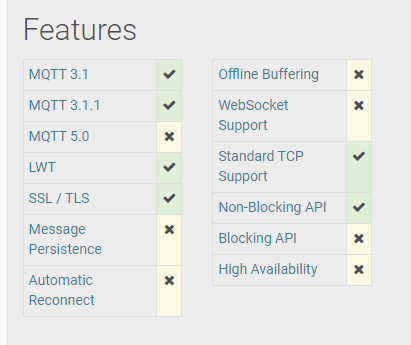
The PhidgetBridge is the interface board needed to measure the output from a load cell. You can connect up to four load cells, strain gauges, or wheatstone bridge sensors.

1. Start with reading the product manual <http://www.phidgets.com/products.php?category=34&product_id=1046_0> and download drivers from: <http://www.phidgets.com/docs/Operating_System_Support> (find your OS and follow the instructions)
2. Next, install the Phidget drivers and connect to them using the Phidget Control Panel
3. Try attaching four sensor to e.g. a chair and see if you can get an accurate weight measurement out of it.

g) Sending Phidget events to MQTT server using Paho M2MQTT

In this step – you will be using the Paho M2MQTT C# .NET client to distribute an event.

M2Mqtt is a MQTT client available for all .Net platforms (.Net Framework, .Net Compact Framework and .Net Micro Framework) and WinRT platforms (Windows 8.1 and Windows Phone 8.1).



Read more on Paho here: <https://www.eclipse.org/paho/index.php?page=clients/dotnet/index.php>

You can either download the Nuget assemeblies – or build them from source.

**Download**

The M2Mqtt client assemblies for using as references in your Visual Studio projects can be downloaded from <https://www.nuget.org/packages/M2Mqtt/> for .NET framework (“classic” Windows .NET) and .NET Core (for Linux and Mac)

**Building from source**

The project can be installed from the repository as well. To do this:

git clone <https://github.com/eclipse/paho.mqtt.m2mqtt.git>

You can open one of the available solutions for Visual Studio in the "org.eclipse.paho.mqtt.m2mqtt" folder

**Documentation**

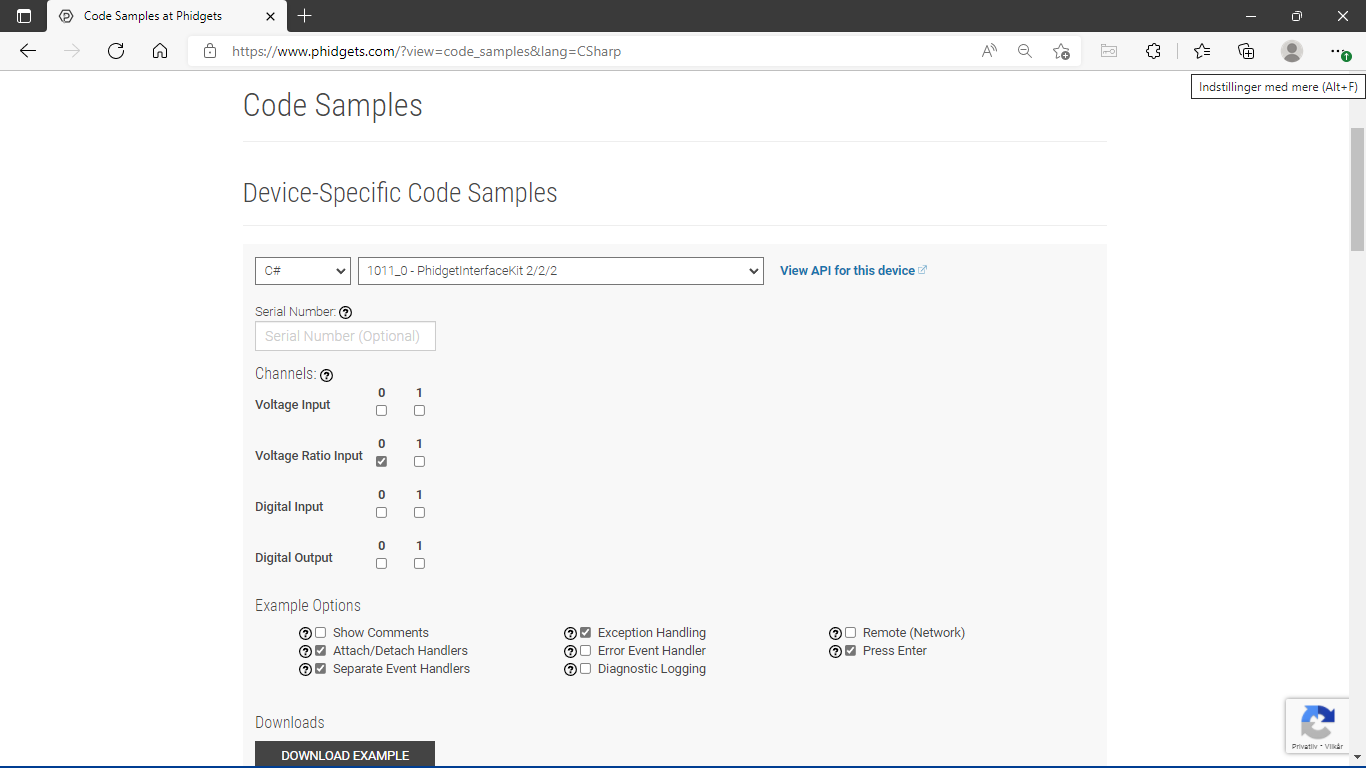
Full client documentation is available on the official M2Mqtt project web site here.

Step 1) Make sure you have Visual Studio 2019 or newer

Step 2) We assume you have installed Phidgets drivers already

Step 3) Consider to use the Sample Builder on Phidgets website

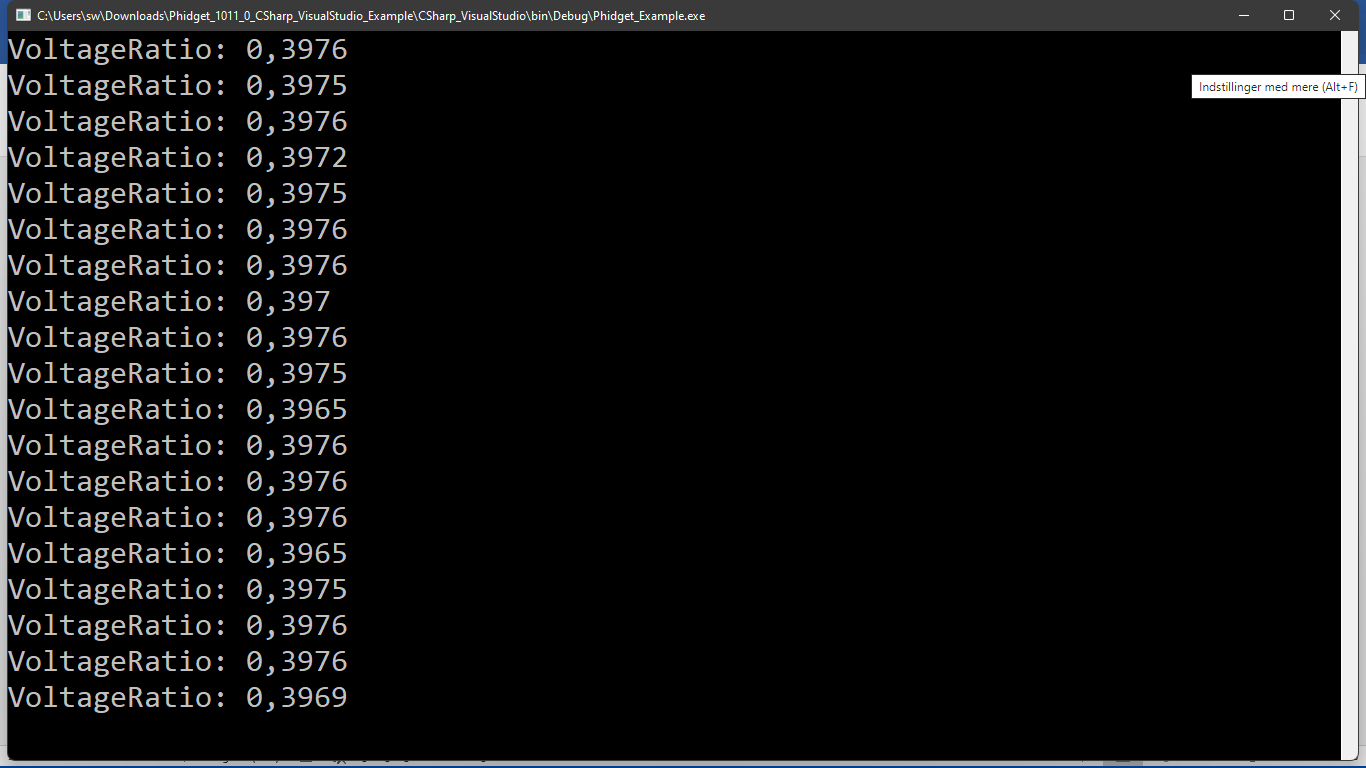
<https://www.phidgets.com/?view=code_samples&lang=CSharp>



Make sure to generate to the correct controller you are using

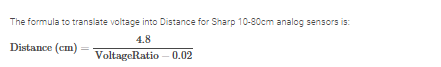


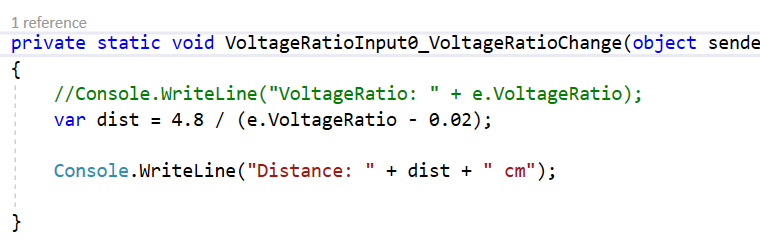
Step 4) Now run the code

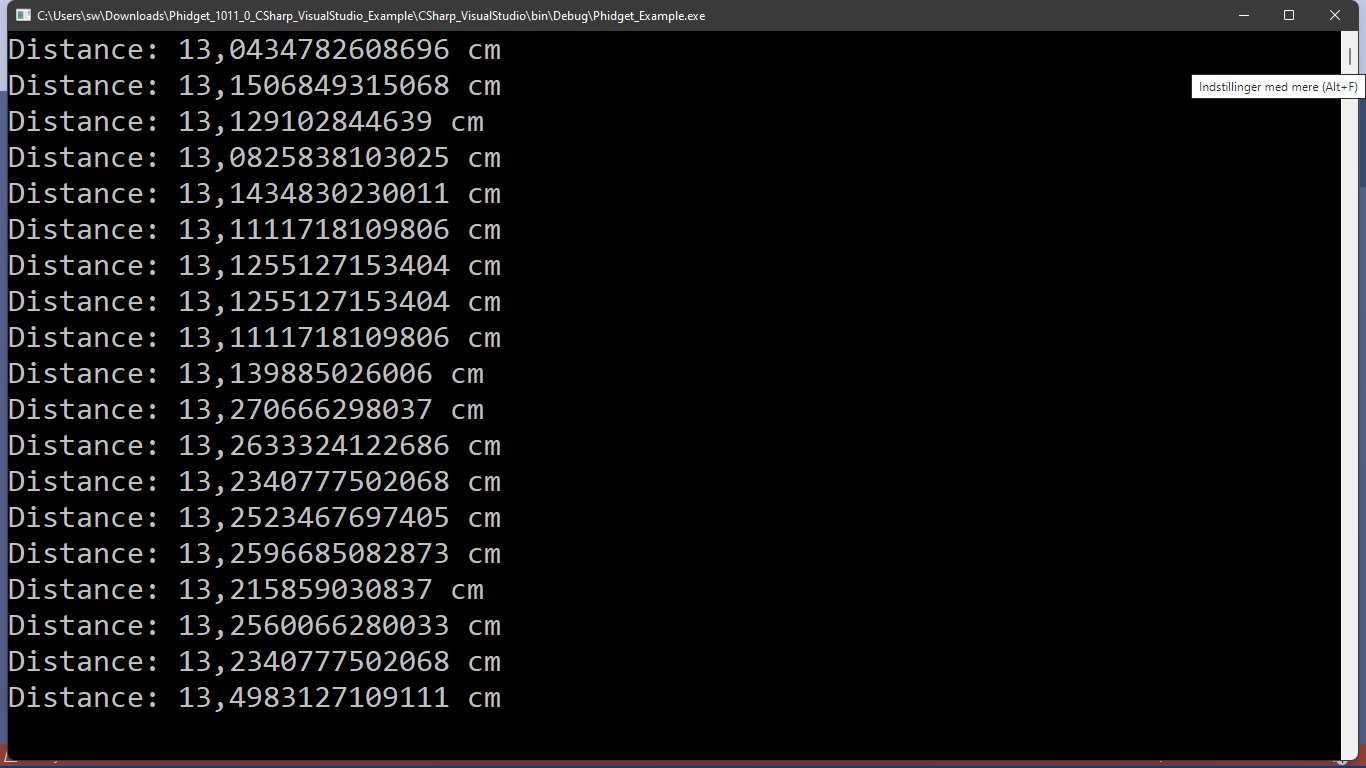


Step 5)

Now – I have attached a Sharp Distance sensor. I want something useful





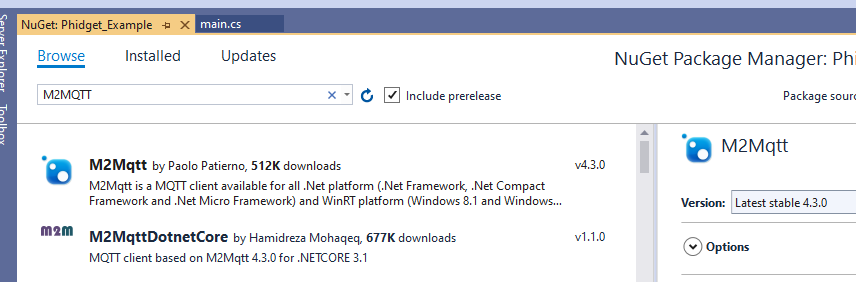


Step 6)

Fine. Now we have a context aware system. But, it is not yet distributed. Let’s distribute it with MQTT.

Ensure you have Mosquitto or other MQTT broker running.

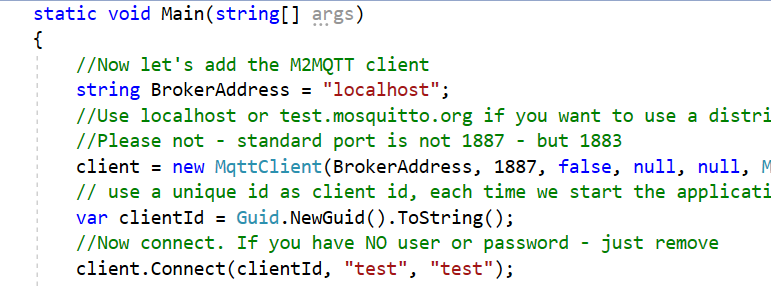
Fetch M2MQTT Nuget package …



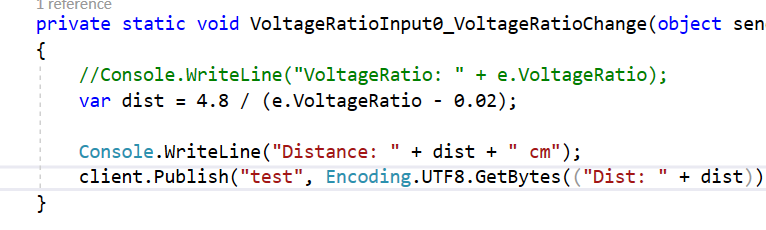
Now to the code



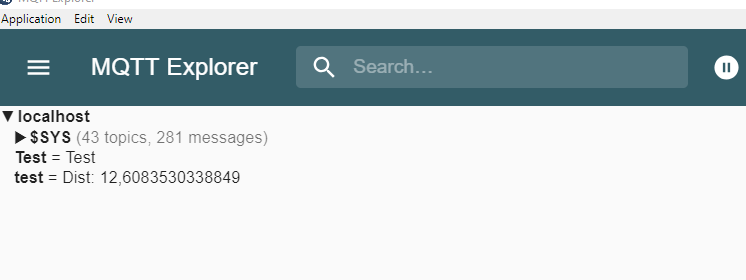
And in the main



And to publish the distance

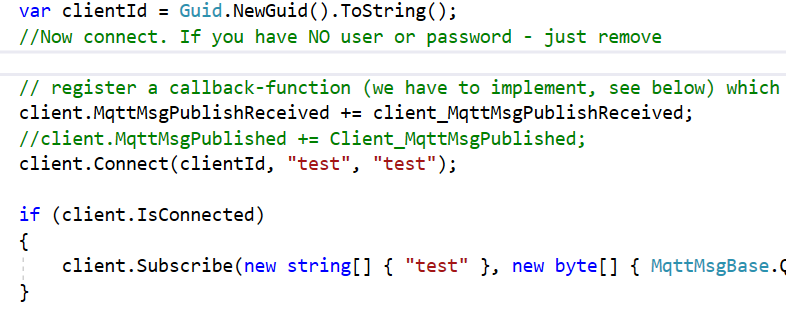


Now run it … and see the following:



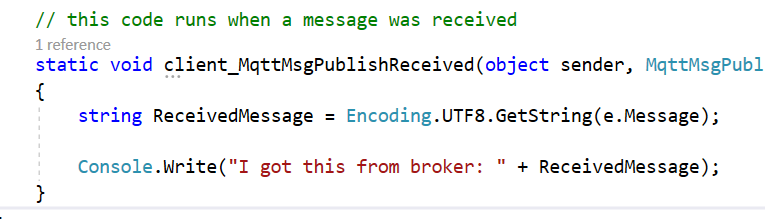
Step 7)

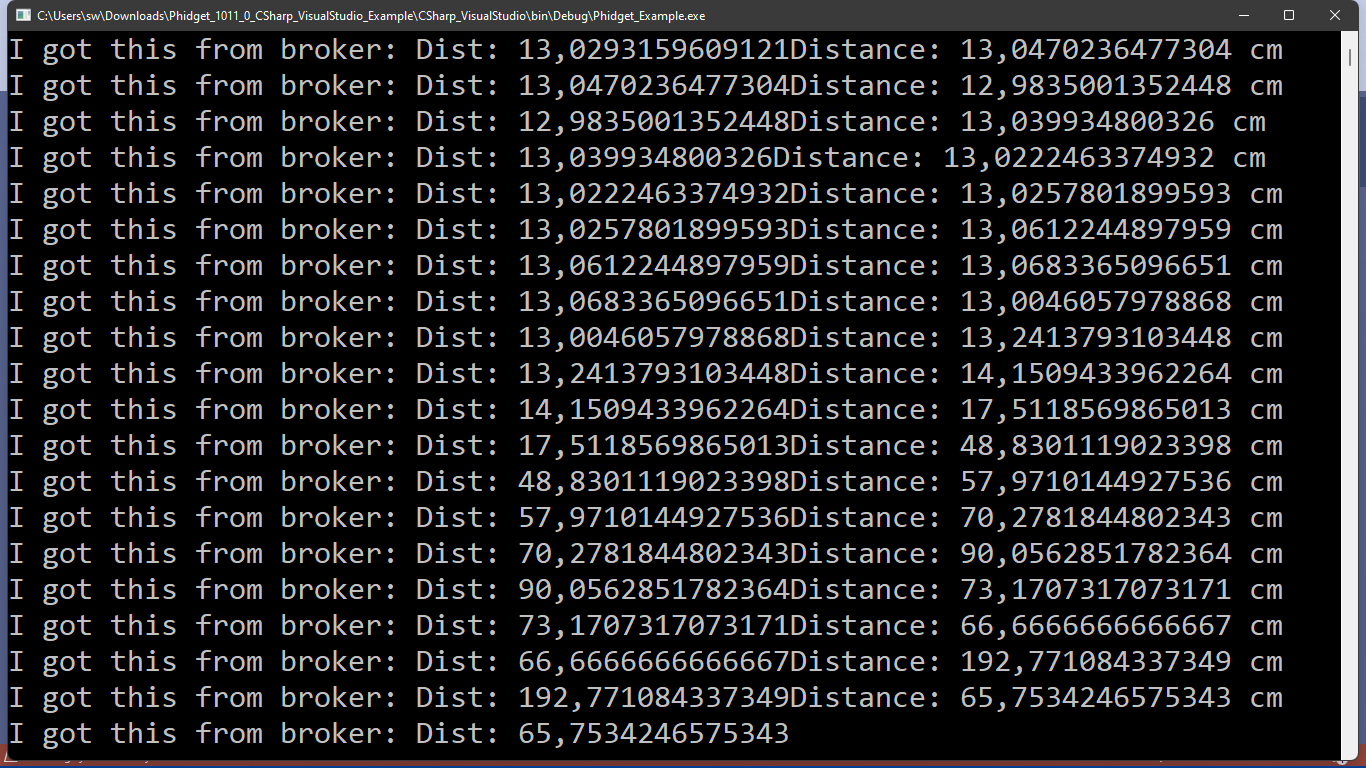
Now – let us also listen after “test” topic events (subscribe) – which is of course something we would want to do in our SERVICE



Here – you can choose to operate at different semantics. At most once, at least once, exactly once.

MqttMsgBase.QOS\_LEVEL\_EXACTLY\_ONCE





Step 8)

Now consider this (discuss in your groups):

1. Now the service is available via all IP enabled networks incl. the internet. How can different programming language use e.g. a Distance Class as a common object? How can they communicate?
2. If we want to make an intelligent environment – how can we change the text format into something that other services could use?
3. How about security? Is this a secure solution?

**h) Building the Pressalit Hygiene Sensor solution with Phidgets**

For this exercise – we are using a Phidgets VINT HUB AD converter along with a capacitive touch sensor to support sensing whether soap is being dispensed. Also, we use two ultrasound sensors to sense a toilet visit and handwashing respectively. Now, neither of these three sensors will guarantee that the action was performed properly, but it they will add to the likelihood of it being so.



Smart mirror giver feedback til deltagerne

Håndvask ”presence” sensor registrerer at håndvasken er benyttet

Toilet ”presence” sensor registrerer et toiletbesøg

Hygiejne sensor registrerer om der tages sæbe





You will need to connect to the VINT HUB via USB cable for the existing setup as shown below.

We use Input 0 for the capacitive touch – and 3 and 4 for the ultra sound sensors



Find the source code under “Source”.

The below snippet is only for showcasing what is needed to initialize the sensors

using System;

using System.Threading;

using System.Threading.Tasks;

using Phidget22;

using uPLibrary.Networking.M2Mqtt;

using uPLibrary.Networking.M2Mqtt.Messages;

using OpenCare.EVODAY;

using Phidget\_Example;

using System.Security.Cryptography.X509Certificates;

using System.Net.Security;

using System.Text;

namespace ConsoleApplication

{

class Program

static void Main(string[] args)

{

Log.Enable(LogLevel.Info, "phidgetlog.log");

DistanceSensor distanceSensor0 = new DistanceSensor();

distanceSensor0.HubPort = 3;

//distanceSensor0.DistanceChange += DistanceSensor0\_DistanceChange;

distanceSensor0.Attach += DistanceSensor0\_Attach;

distanceSensor0.Detach += DistanceSensor0\_Detach;

distanceSensor0.Error += DistanceSensor0\_Error;

DistanceSensor distanceSensor1 = new DistanceSensor();

distanceSensor1.HubPort = 4;

//distanceSensor0.DistanceChange += DistanceSensor0\_DistanceChange;

distanceSensor1.Attach += DistanceSensor0\_Attach;

distanceSensor1.Detach += DistanceSensor0\_Detach;

distanceSensor1.Error += DistanceSensor0\_Error;

//VoltageRatioSensorType voltageRatioSensor = new VoltageRatioSensorType();

VoltageRatioInput voltageRatioInput = new VoltageRatioInput();

voltageRatioInput.HubPort = 0;

voltageRatioInput.IsHubPortDevice = true;

voltageRatioInput.Channel = 0;

voltageRatioInput.VoltageRatioChange += VoltageRatioInput\_VoltageRatioChange;

try

{

InitizlizeMqttClient();

distanceSensor0.Open(5000);

distanceSensor0.DataInterval = 2000;

distanceSensor0.SonarQuietMode = false;

distanceSensor1.Open(5000);

distanceSensor1.DataInterval = 2000;

distanceSensor1.SonarQuietMode = false;

voltageRatioInput.Open(5000);

//voltageRatioInput.SensorType = VoltageRatioSensorType.PN\_1129;

voltageRatioInput.DataInterval = 100;

Next, we send the HEUCOD event.

private static void SendHandwashingEvent()

{

var sensorData = new OpenCare.EVODAY.EDL.HandWashingEvent();

sensorData.SensorId = "PressalitHandWash1" + Setting.Default.PatientID;

sensorData.Value = 1;

sensorData.PatientId = Setting.Default.PatientID;

sensorData.MonitorId = Setting.Default.PatientID;

sensorData.Description = "Pressalit hand wash tracker";

sensorData.DeviceModel = "SoapDetect";

sensorData.DeviceVendor = "Pressalit";

sensorData.Room = "Bathroom";

sensorData.Timestamp = Utils.ConvertToUnixTime(DateTime.UtcNow);

sensorData.StartTime = Utils.ConvertToUnixTime(DateTime.UtcNow);

sensorData.EndTime = Utils.ConvertToUnixTime(DateTime.UtcNow);

var message = OpenCare.EVODAY.Serialize.ToJson((BasicEvent)sensorData);

SendMQTT(message);

}

Likewise, we use the capacitative touch sensor for the soap dispenser.

private static void VoltageRatioInput\_VoltageRatioChange(object sender, Phidget22.Events.VoltageRatioInputVoltageRatioChangeEventArgs e)

{

if (lastSoapDispensingEventSent == null || (DateTime.Now -  (DateTime)lastSoapDispensingEventSent).TotalSeconds > 300)

if (e.VoltageRatio > 0.9)

{

Console.WriteLine("Soap dispensed");

lastSoapDispensingEventSent = DateTime.Now;

lastHandWashEventSent = null;

SendSoapDispensnig();

}

}