Nanyang Polytechnic

School of Information Technology

Course	:	Diploma in Infocomm & Security (ITDF12)
Module	:	Sensor Technologies and Project (ITP272)

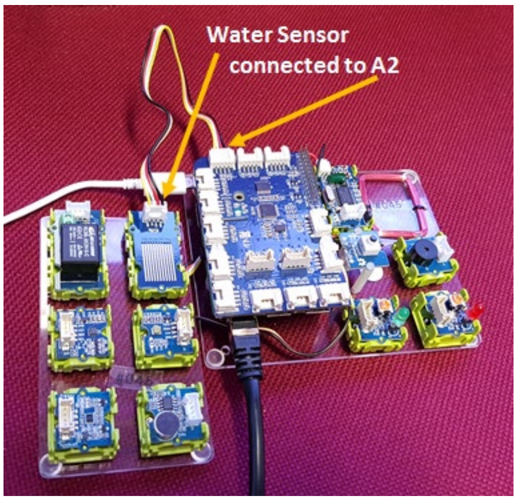
Raspberry Pi Practical : Programming Raspberry Pi with Grove Buzzer

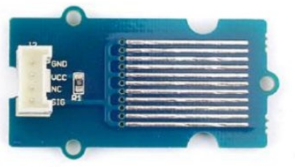
Objectives:

- Learn how to interface with Analog I/O port
- Learn how to create program to control Grove Water Sensor.
- Learn how to create program with state machines to work with Water Sensor, buzzer and push button

Exercise 1: Creating WaterDetect

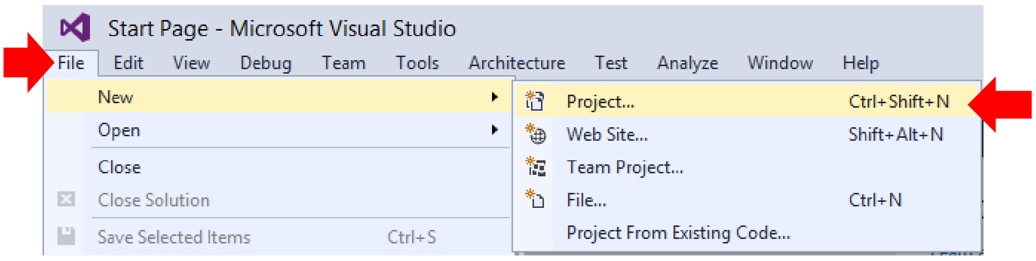
1. Today we will be creating a program that will measure the presence of water ponding on the sensor.
2. We shall use Digital Input port **D8** of the GrovePi for the Ultrasonic Distance Sensor. Ensure that you have port **D8** connected to the **Ultrasonic Sensor**.



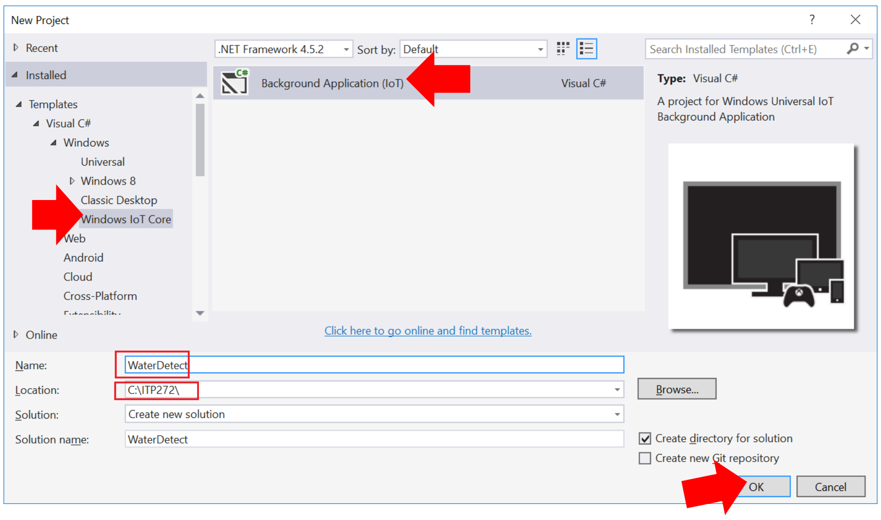


Water Sensor

3. Start Visual Studio 2015, Click on File – New - Project.



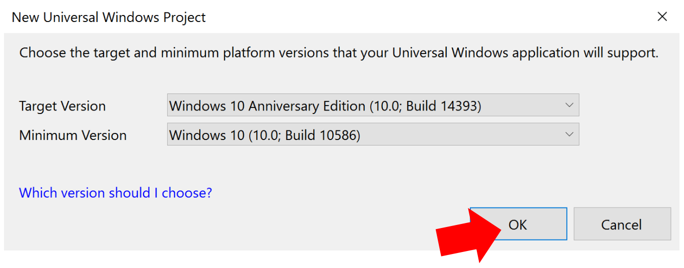
4. From the Installed Templates, select Visual C# – Windows IoT Core – Background Application. Name your project as **WaterDetect**, save it in you ITP272 folder and Click OK.



Type: Visual C#
A project for Windows Universal IoT Background Application

Name: WaterDetect
Location: C:\ITP272\
Solution: Create new solution
Solution name: WaterDetect
[✓] Create directory for solution
[ ] Create new Git repository

5. You should see this pop-up. Click OK.


New Universal Windows Project

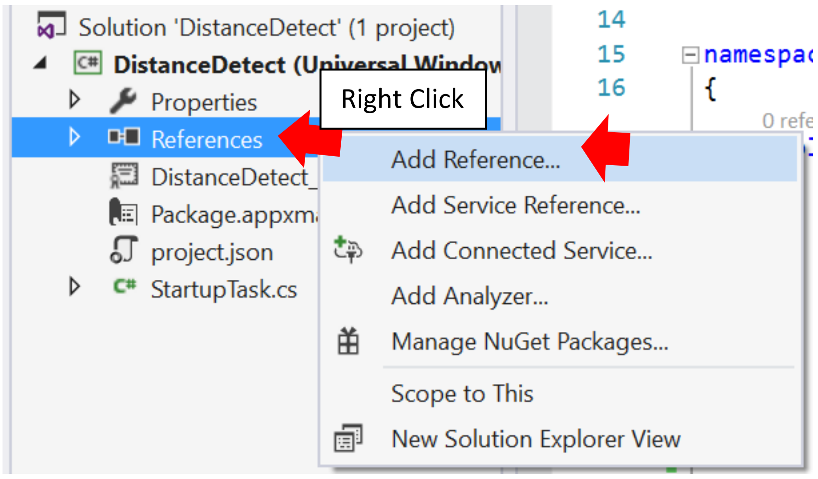
Choose the target and minimum platform versions that your Universal Windows application will support.

Target Version: Windows 10 Anniversary Edition (10.0; Build 14393)

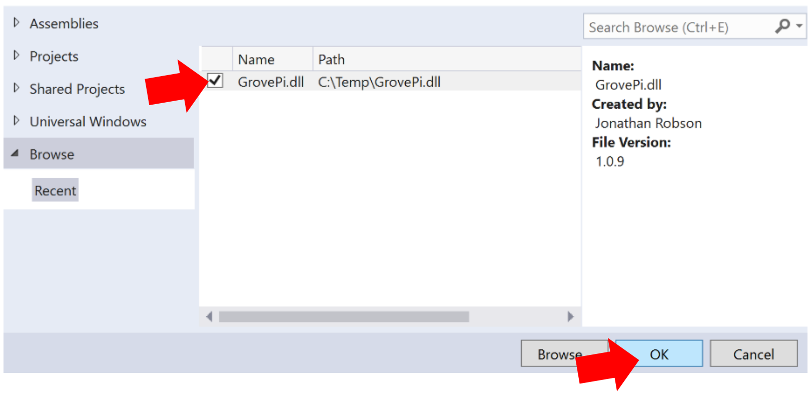
Minimum Version: Windows 10 (10.0; Build 10586)

6. At the solution explorer on the right side of the screen, check that you have StartupTask.cs. This is the startup program file. Double Click on Startup Task.cs.

7. Add Reference for the GrovePi. Right Click on References on Solution Explorer and Click on Add Reference.



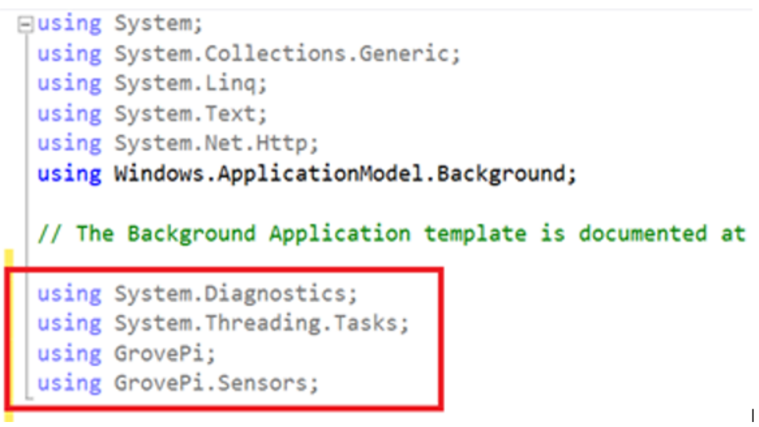
8. Download and select GrovePi.dll as usual. It should appear in your Recent if you've used it before. Check on it and Click OK.



	Name	Path
✓	GrovePi.dll	C:\Temp\GrovePi.dll

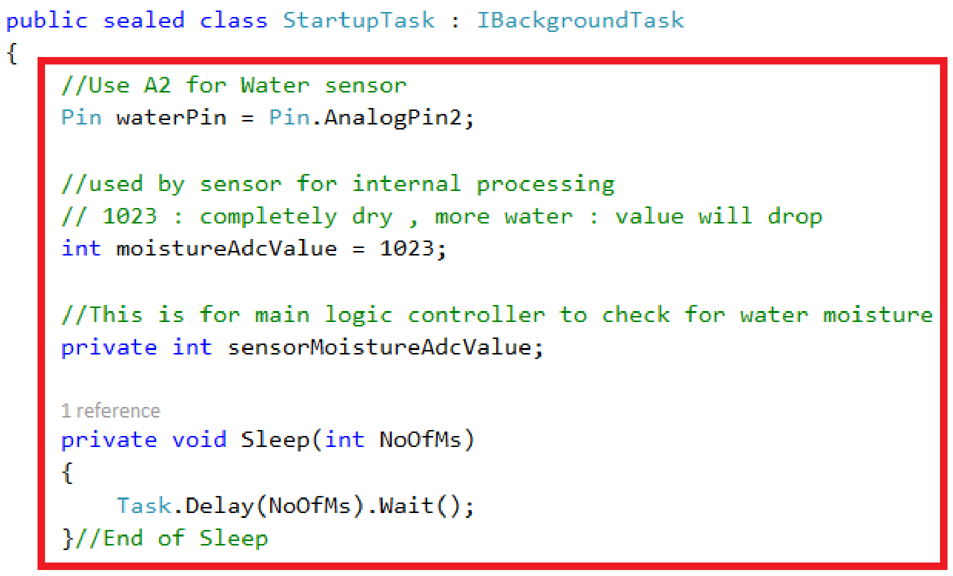
Name: GrovePi.dll
Created by: Jonathan Robson
File Version: 1.0.9

9. Go to your "StartupTask.cs" and type in the following codes above your namesapce to include required libraries. It is fine to be in gray initially since you have not used them in the codes



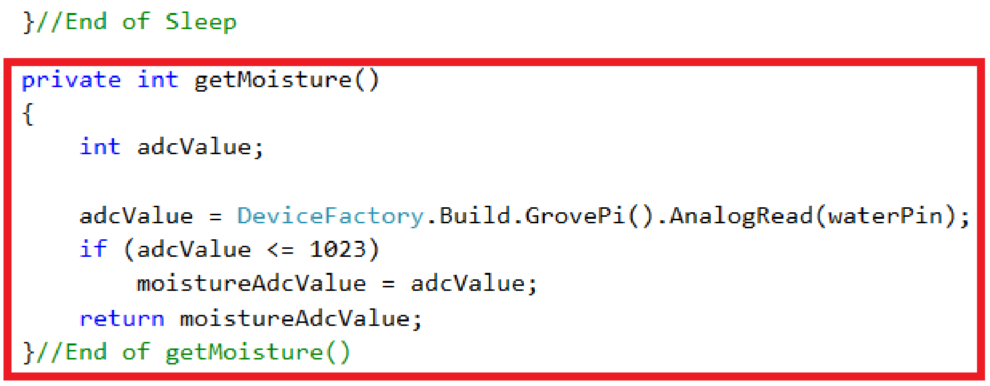
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Http;
using Windows.ApplicationModel.Background;

// The Background Application template is documented at

using System.Diagnostics;
using System.Threading.Tasks;
using GrovePi;
using GrovePi.Sensors;
```

10. Add in these codes in the StartupTask.cs



```
public sealed class StartupTask : IBackgroundTask
{
    //Use A2 for Water sensor
    Pin waterPin = Pin.AnalogPin2;

    //used by sensor for internal processing
    // 1023 : completely dry , more water : value will drop
    int moistureAdcValue = 1023;

    //This is for main logic controller to check for water moisture
    private int sensorMoistureAdcValue;

    1 reference
    private void Sleep(int NoOfMs)
    {
        Task.Delay(NoOfMs).Wait();
    }//End of Sleep
```

11. Create the **getMoisture()** method below Sleep(). This method reads the moisture and update the **moistureAdcVaue** variable.



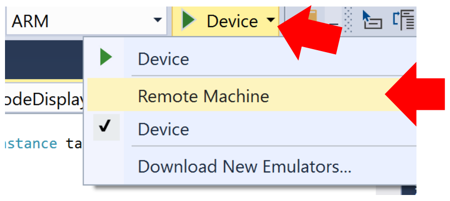
```
}//End of Sleep

private int getMoisture()
{
    int adcValue;

    adcValue = DeviceFactory.Build.GrovePi().AnalogRead(waterPin);
    if (adcValue <= 1023)
        moistureAdcValue = adcValue;
    return moistureAdcValue;
}//End of getMoisture()
```

The method use the **AnalogRead()** method to read in the sensor value.

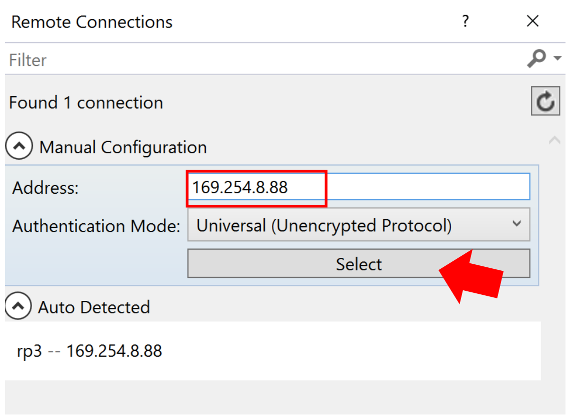
12. Finally, add the codes in the **Run()** method.

```
public void Run(IBackgroundTaskInstance taskInstance)
{
    //
    // TODO: Insert code to perform background work
    //
    // If you start any asynchronous methods here, prevent the task
    // from closing prematurely by using BackgroundTaskDeferral as
    // described in http://aka.ms/backgroundtaskdeferral
    //

    while (true)
    {
        Sleep(300);
        sensorMoistureAdcValue = getMoisture(); //get moisture from sensor
        Debug.WriteLine("Moisture = " + sensorMoistureAdcValue);
        if(sensorMoistureAdcValue > 1000)
            Debug.WriteLine("Dry");
        else if (sensorMoistureAdcValue < 100)
            Debug.WriteLine("There is water ponding");
        else
            Debug.WriteLine("Moderately wet");
    }
}
```

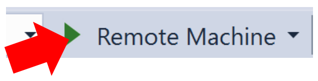
13. In order to deploy the program to your hardware, click on the Drop Down button beside the Device and Select Remote Machine.



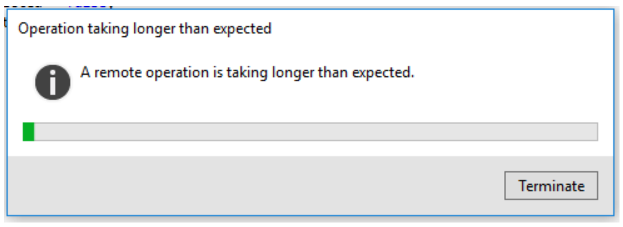
14. Key in the address manually like shown below. Click on **Select** after that.


Remote Connections
Filter
Found 1 connection
Manual Configuration
Address: 169.254.8.88
Authentication Mode: Universal (Unencrypted Protocol)
Select
Auto Detected
rp3 -- 169.254.8.88

15. Click on Remote Machine to Run the Program.



16. You may see the following warning but it is fine. Every time when you deploy a project for the first time on the hardware, it will take a long time to deploy. Just wait for a while and it should be deployed successfully.

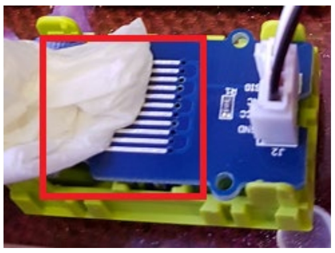


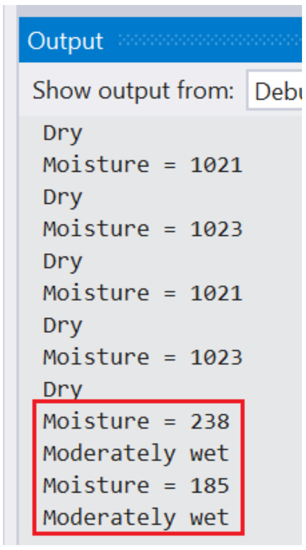
Operation taking longer than expected

A remote operation is taking longer than expected.

17. Upon successful deployment, you should see the distance reported at the Output windows.

Initially when the sensor is dry, your should see a value that is > 1000. Take a wet tissue and pressed it GENTLY on the water sensor. You should see the value drop.





```
Dry
Moisture = 1021
Dry
Moisture = 1023
Dry
Moisture = 1021
Dry
Moisture = 1023
Dry
Moisture = 238
Moderately wet
Moisture = 185
Moderately wet
```

You've just learnt how to get the moisture level from the Water sensor.
In the **Run**() method, once you call the **getMoisture** () method to read in the moisture from the sensor and save the moisture in the **sensorMoistureAdcValue** variable. The **Run**() method can check this **sensorMoistureAdcValue** variable to see moisture level on the sensor.

Exercise 2: Creating a WaterLevelMonitoringSystem

Implement a Flood monitoring system. The system has a water sensor placed at the side of the drain that could detect whether water level is too high in the drain. The system is configured with 2 safety water levels. A Warning level and an Alarm level. Under normal circumstances, all indicator lights are off. When the sensor is slightly detecting moisture, it goes into warning mode and shows green indicator light. When more of the sensor is detecting moisture, it goes into alarm mode and shows blinking red indicator light. This system helps to monitor the water level in drains to notify when there is impending flooding.

To design a system that meets the above criteria, we can design a state machine to function as required.

We've going to write the codes such that your program will behave as below. Show your tutor when you're done coding and verifying.

Your program starts in **MODE_Normal**

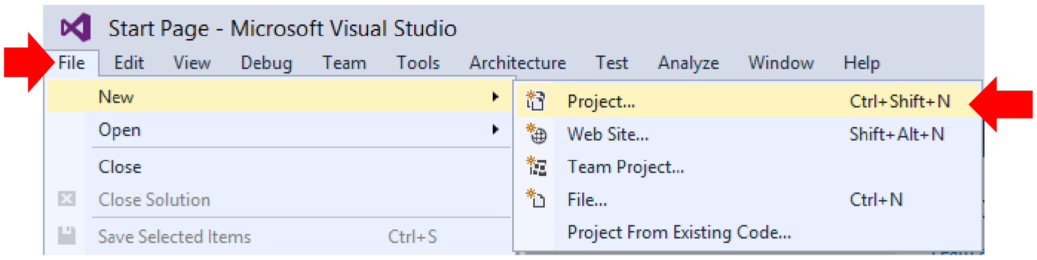
- **MODE_Normal**
    - Display moisture level and turns off all LED
    - If water touches the sensor, change mode to **MODE_WARNING**
    - If water level rises causing sensor to be exposed to more water, change mode to **MODE_ALARM**

- **MODE_WARNING**
    - Turns on the Green LED and turns off the Red LED
    - If water level rises causing sensor to be exposed to more water, change mode to **MODE_ALARM**
    - If water level drop and sensor is no longer sensing water, change mode to back **MODE_Normal**

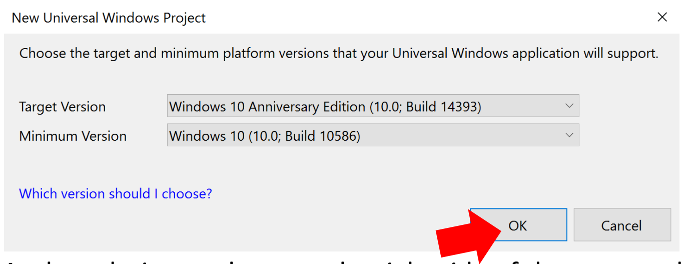
- **MODE_ALARM**
    - Turns off Green LED and make Red LED blinking
    - If water touches the sensor, change mode to **MODE_WARNING**
    - If water level drop and sensor is no longer sensing water, change mode to back **MODE_Normal**

You can try to do this on your own to test your own understanding so that you can write you own program for your project later. The next pages shows the solution to this and you could use it to verify and see where you go wrong. Check with your tutor when in doubt.

1. In this exercise, we will be creating a program **WaterLevelMonitoringSystem** as stated in the previous page.
2. Add in Green and Red LED. Ensure that you have the following connections
   - A2 : Water Sensor
   - D5 : Red LED
   - D6 : Green LED
3. Start Visual Studio 2015, Click on File – New - Project.



4. From the Installed Templates, select Visual C# – Windows IoT Core – Background Application. Name your project as **WaterLevelMonitoringSystem**, save it in you ITP272 folder and Click OK.
5. You should see this pop-up. Click OK.



New Universal Windows Project

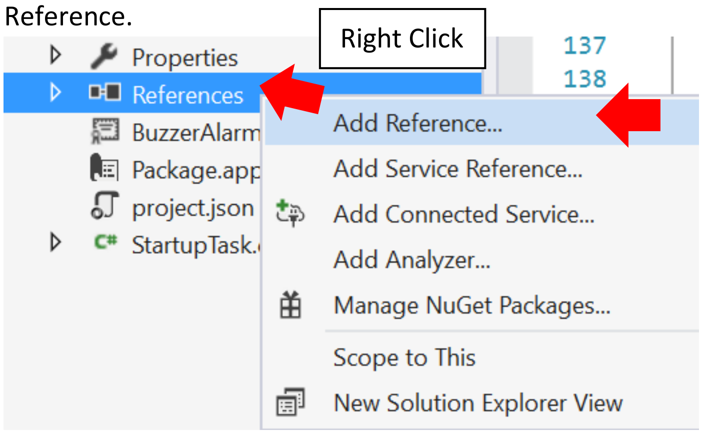
Choose the target and minimum platform versions that your Universal Windows application will support.

Target Version: Windows 10 Anniversary Edition (10.0; Build 14393)

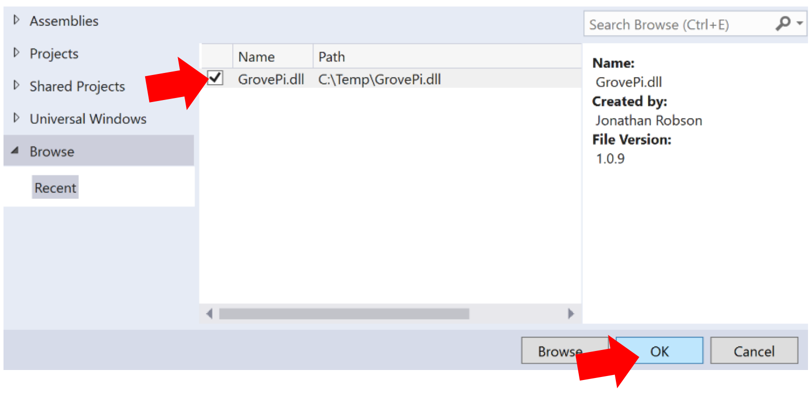
Minimum Version: Windows 10 (10.0; Build 10586)

Which version should I choose?

6. At the solution explorer on the right side of the screen, check that you have StartupTask.cs. This is the startup program file. Double Click on Startup Task.cs.
7. Add Reference. Right Click on References on Solution Explorer and Click on Add Reference.



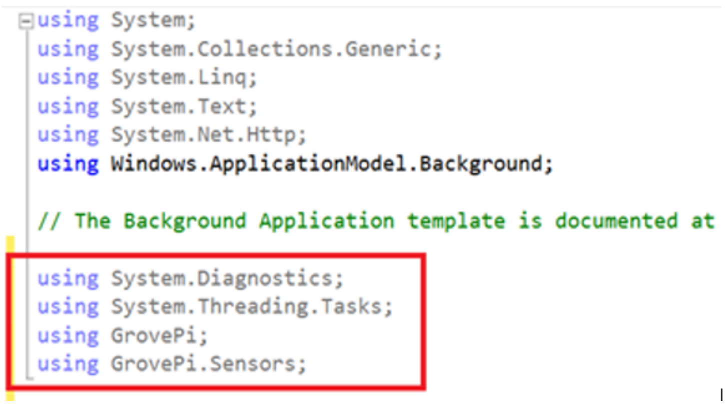
8. Select GrovePi.dll as usual



Name	Path
GrovePi.dll	C:\Temp\GrovePi.dll

Name:
GrovePi.dll
Created by:
Jonathan Robson
File Version:
1.0.9

9. Go to your "StartupTask.cs" and type in the following codes above your namesapce to include required libraries. It is fine to be in gray initially since you have not used them in the codes



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Http;
using Windows.ApplicationModel.Background;

// The Background Application template is documented at

using System.Diagnostics;
using System.Threading.Tasks;
using GrovePi;
using GrovePi.Sensors;
```

10. Add in these codes in the StartupTask.cs

```
public sealed class StartupTask : IBackgroundTask
{
    //State Machine vairables to control different mode of operation
    const int MODE_NORMAL = 1;
    const int MODE_WARNING = 2;
    const int MODE_ALARM = 3;
    static int curMode;

    //Use A2 for Water sensor, Use D5 for Red LED, D6 for Green LED,
    Pin waterPin = Pin.AnalogPin2;
    ILed ledRed = DeviceFactory.Build.Led(Pin.DigitalPin5);
    ILed ledGreen = DeviceFactory.Build.Led(Pin.DigitalPin6);

    //used by sensor for internal processing
    // 1023 : completely dry , more water : value will drop
    int moistureAdcValue = 1023;

    //This is for main logic controller to check for water moisture
    private int sensorMoistureAdcValue;

    private void Sleep(int NoOfMs)
    {
        Task.Delay(NoOfMs).Wait();
    }//End of Sleep
```

11. Create the method **getMoisture()** below the Sleep().

```
}//End of Sleep
```

```
private int getMoisture()
{
    int adcValue;

    adcValue = DeviceFactory.Build.GrovePi().AnalogRead(waterPin);
    if (adcValue <= 1023)
        moistureAdcValue = adcValue;
    return moistureAdcValue;
}//End of getMoisture()
```

12. Create a handler methods for the operation of the various modes.
Add the below codes to handle the Normal mode.

```
}//End of getMoisture()
```

```
private void handleModeNormal()
{
    //  1.  Define Behaviour in this mode
    ledRed.ChangeState(SensorStatus.Off);
    ledGreen.ChangeState(SensorStatus.Off);

    // 2.   Must write the condition to move on to other modes
    if (sensorMoistureAdcValue < 200)
    {
        //Move on to Mode Alarm when sensor exposed to more water
        curMode = MODE_ALARM;
        Debug.WriteLine("===Entering  MODE_ALARM===");
    }
    else if(sensorMoistureAdcValue < 1000)
    {
        //Move on to Mode Warning when sensor exposed to little water
        curMode = MODE_WARNING;
        Debug.WriteLine("===Entering  MODE_WARNING===");
    }
}//End of handleModeNormal()
```

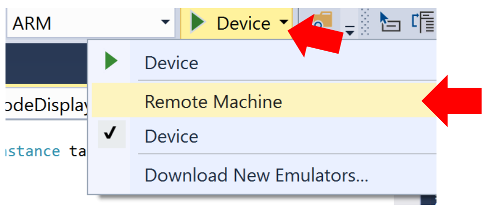
13. Further add the following codes for Warning mode ad Alarm mode

```
}//End of handleModeNormal()
private void handleModeWarning()
{
    //  1.  Define Behaviour in this mode
    ledRed.ChangeState(SensorStatus.Off);
    ledGreen.ChangeState(SensorStatus.On);

    // 2.   Must write the condition to move on to other modes
    if (sensorMoistureAdcValue < 200)
    {
        //Move on to Mode Alarm when sensor exposed to more water
        curMode = MODE_ALARM;
        Debug.WriteLine("===Entering  MODE_ALARM===");
    }
    else if (moistureAdcValue >= 1000)
    {
        //Move on to Mode Normal when sensor not exposed to water
        curMode = MODE_NORMAL;
        Debug.WriteLine("===Entering  MODE_NORMAL===");
    }
}//End of handleModeWarning()

private void handleModeAlarm()
{
    //  1.  Define Behaviour in this mode
    ledGreen.ChangeState(SensorStatus.Off);
    ledRed.ChangeState(SensorStatus.On);  Sleep(1000);
    ledRed.ChangeState(SensorStatus.Off); Sleep(1000);

    // 2.   Must write the condition to move on to other modes
    if (sensorMoistureAdcValue >= 200)
    {
        //Move on to Mode Warning when sensor exposed to little water
        curMode = MODE_WARNING;
        Debug.WriteLine("===Entering  MODE_WARNING===");
    }
    else if (sensorMoistureAdcValue >= 1000)
    {
        //Move on to Mode Normal when sensor not exposed to water
        curMode = MODE_NORMAL;
        Debug.WriteLine("===Entering  MODE_NORMAL===");
    }
}//End of handleModeAlarm()
```

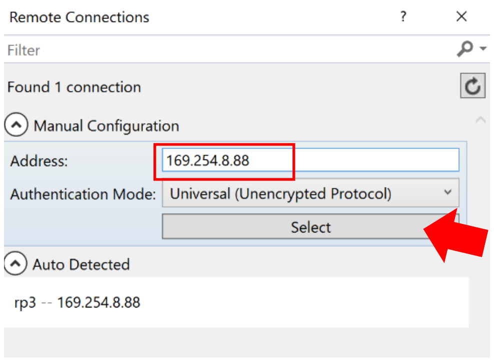
14. Lastly, add the codes for the **Run()** method.

```
public void Run(IBackgroundTaskInstance taskInstance)
{
    //
    // TODO: Insert code to perform background work
    //
    // If you start any asynchronous methods here, prevent the task
    // from closing prematurely by using BackgroundTaskDeferral as
    // described in http://aka.ms/backgroundtaskdeferral
    //

    //Init Mode
    curMode = MODE_NORMAL;
    ledRed.ChangeState(SensorStatus.Off);
    Debug.WriteLine("===Entering  MODE_NORMAL===");

    //This makes sure the main program runs indefinitely
    while (true)
    {
        Sleep(300);
        sensorMoistureAdcValue = getMoisture(); //get moisture from sensor
        Debug.WriteLine("Moisture = " + sensorMoistureAdcValue);

        //State machine handling
        if (curMode == MODE_NORMAL)
            handleModeNormal();
        else if (curMode == MODE_WARNING)
            handleModeWarning();
        else if (curMode == MODE_ALARM)
            handleModeAlarm();
        else
            Debug.WriteLine("ERROR: Invalid Mode. Please check your logic");
    }
}
```

15. Click on the Drop Down button beside the Device and Select "**Remote Machine**" to configure the deployment settings

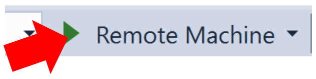


16. Key in the address "**169.254.8.88**" manually as shown below and click on "**Select**" after that. (if you didn't see this screen, refer to **Practical 1 Exercise 2** on steps to display this screen).

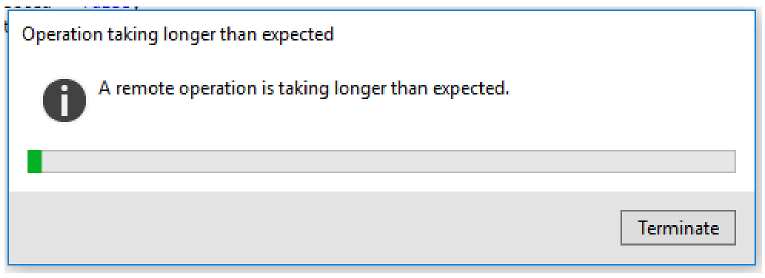


Remote Connections
Filter
Found 1 connection
Manual Configuration
Address: 169.254.8.88
Authentication Mode: Universal (Unencrypted Protocol)
Select
Auto Detected
rp3 -- 169.254.8.88

17. Click on Remote Machine to Run the Program.



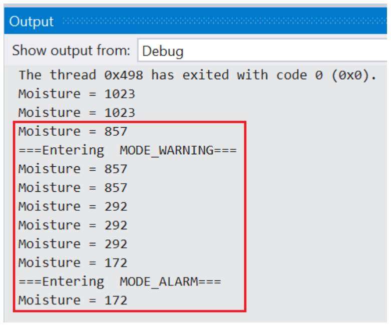
18. You may see the following warning but it is fine. Every time when you deploy a project for the first time on the hardware, it will take a long time to deploy. Just wait for a while and it should be deployed.



Operation taking longer than expected
A remote operation is taking longer than expected.
Terminate

19. Once deployed successfully, the program will start at MODE_NORMAL. Both LED should be off. Use a wet tissue to touch the edge of the water sensor. You should see it goes to Warning mode where the Green LED will turn on.

Use the wet tissue to touch more of the sensor (you can also make the tissue wetter) and you should see the mode change to alarm mode with red LED blinking.



Output
Show output from: Debug

```
The thread 0x498 has exited with code 0 (0x0).
Moisture = 1023
Moisture = 1023
Moisture = 857
===Entering  MODE_WARNING===
Moisture = 857
Moisture = 857
Moisture = 292
Moisture = 292
Moisture = 292
Moisture = 172
===Entering  MODE_ALARM===
Moisture = 172
```

==End of Practical==