



School of Information Technology

Course : Diploma in Infocomm & Security (ITDF12)

Module : Sensor Technologies and Project (ITP272)

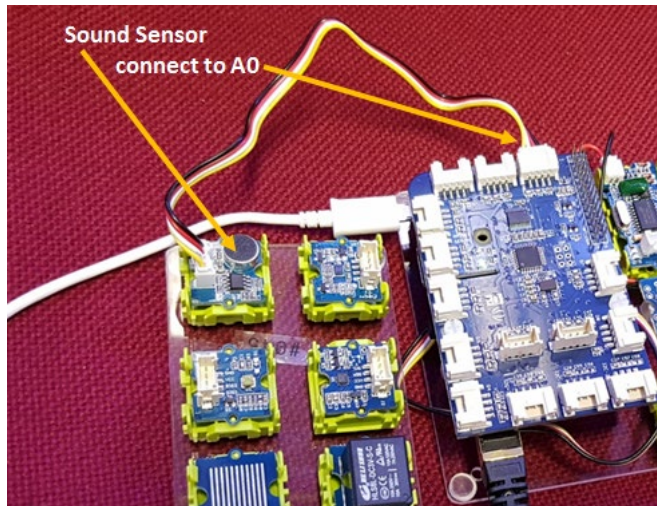
Raspberry Pi Practical : Programming Raspberry Pi with Grove Sound Sensor

Objectives:

- Learn how to interface with Analog Input and use it read in sound sensor data
- Learn how to create program to interface with the Sound Sensor.

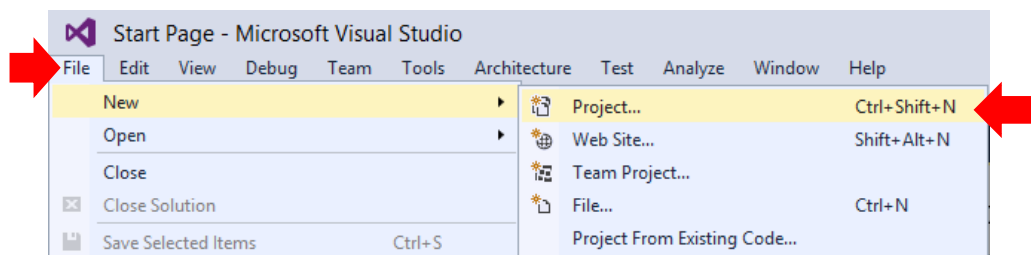
Exercise 1: Creating SoundDetect

1. Today we will be creating a program that will read data from Sound sensor (Mic).
2. We shall use Analog port **A0** of the GrovePi for the Sound. Ensure that you have port **A0** connected to the **Sound Sensor**.



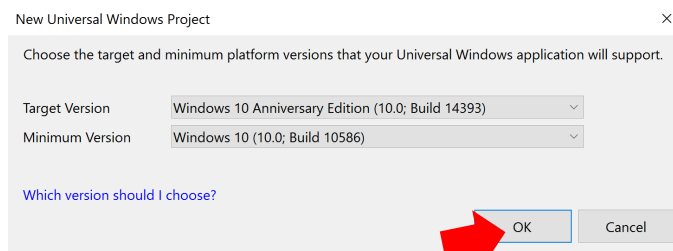
Sound Sensor

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



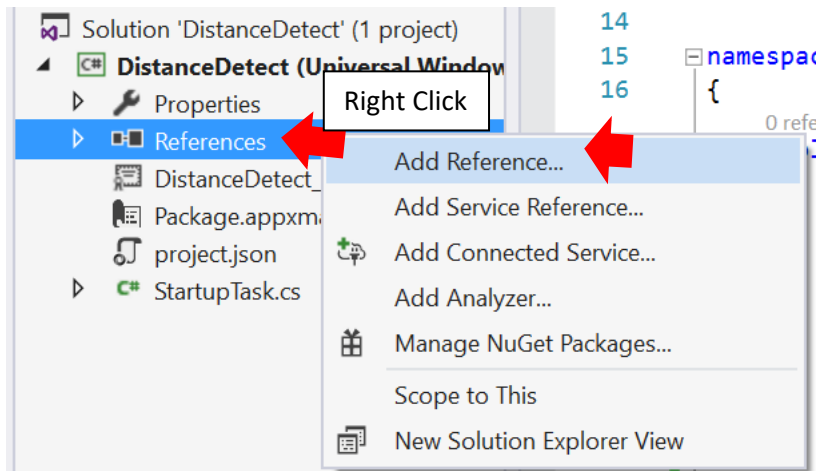
4. From the Installed Templates, select Visual C# – Windows IoT Core – Background Application. Name your project as **SoundDetect**, save it in you ITP272 folder and Click OK.

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

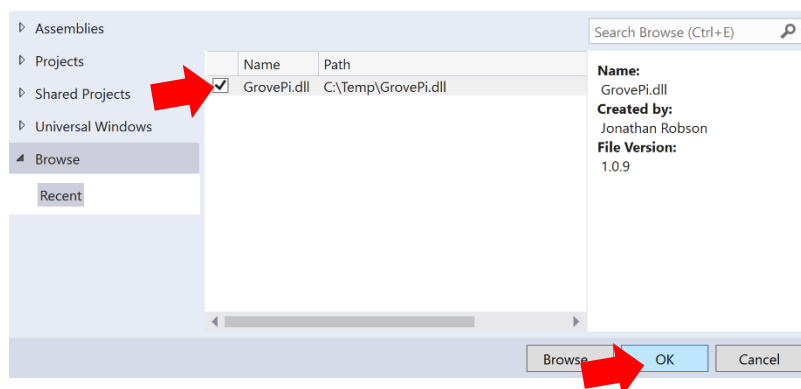


6. At the solution explorer on the right side of the screen, check that you have StartupTask.cs. This is the startup program file. Double Click on Startup Task.cs.

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.



9. Add in the following namespace codes

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

// The Background Application templ

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

namespace SoundDetect
```

10. Add in these codes in the StartupTask.cs

```
public sealed class StartupTask : IBackgroundTask
{
    //Use A0 for sound sensor (mic)
    Pin soundSensor = Pin.AnalogPin0;

    //This is for main logic controller to know that sound has been detected
    private static int sesnsorSound = 0;

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

11. Add in the codes to detect environment sound levels.

```
    }//End Sleep()
```

```
private int getSound()
{
    int sum = 0;
    int count = 0;
    int sound = 0;
    while (++count <= 10)
    {
        Sleep(10);
        sound = DeviceFactory.Build.GrovePi().AnalogRead(soundSensor);
        if (sound < 1024)
            sum += sound;
        else
            count--;
    }
    return sum;
}
} //End of getSound()
```

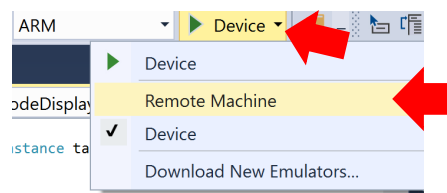
12. Finally, add the codes in the **Run()** method.

```
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
    //
    int count = 0;

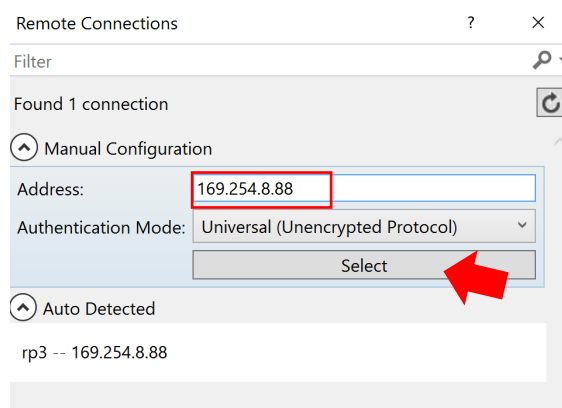
    Debug.WriteLine("\n\nSound Detect program started");
    Debug.WriteLine("Please note that sound sensor is very sensitive");
    Debug.WriteLine("to environment noises. ");
    Debug.WriteLine("Environment nominal noise level varies a lot depending on");
    Debug.WriteLine("\t - different times of day");
    Debug.WriteLine("\t - different location");
    Debug.WriteLine("It may pick up Inaudible sound signal as well!");
    Debug.WriteLine("You may not easily use it reliably in your project!");
    Debug.WriteLine("Use it at your own risk!\n\n");
    Sleep(9000);
    Debug.WriteLine("Test now");

    while (true)
    {
        Sleep(300);
        sesnsorSound = getSound();
        Debug.WriteLine(count++ + " Nominal Sound Value = " + sesnsorSound);
    }
}
```

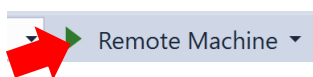
13. In order to deploy the program to your hardware, click on the Drop Down button beside the Device and Select Remote Machine.



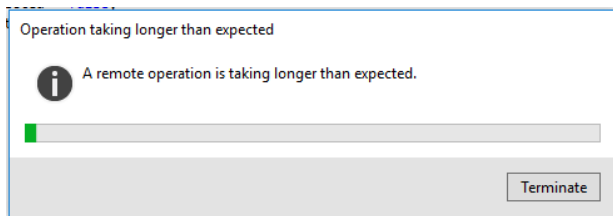
14. Key in the address manually like shown below. Click on **Select** after that.



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.



17. Upon successful deployment, you should see values reported by the Sound Sensor. Note that these values are showing the current noise level of your environment. You may need to always amend your codes to compensate for these environment sound for your project to work correctly. Please note that comments here. It is easy to have false triggering for sound sensor due to background noise and thus it may not be easy for you to use the sound sensor reliably.

A screenshot of the 'Output' window in Visual Studio, with the 'Debug' tab selected. The output text is as follows:

```
Output
Show output from: Debug

Sound Detect program started
Please note that sound sensor is very sensitive
to environment noises.
Environment nominal noise level varies a lot depending on
- different times of day
- different location
It may pick up Inaudible sound signal as well!
You may not easily use it reliably in your project!
Use it at your own risk!

'backgroundTaskHost.exe' (CoreCLR: CoreCLR_UWP_Domain): Lo
Test now
0 Nominal Sound Value = 794
1 Nominal Sound Value = 797
2 Nominal Sound Value = 960
3 Nominal Sound Value = 1464
4 Nominal Sound Value = 796
5 Nominal Sound Value = 794
6 Nominal Sound Value = 796
The thread 0xb20 has exited with code 0 (0x0).
7 Nominal Sound Value = 799
```

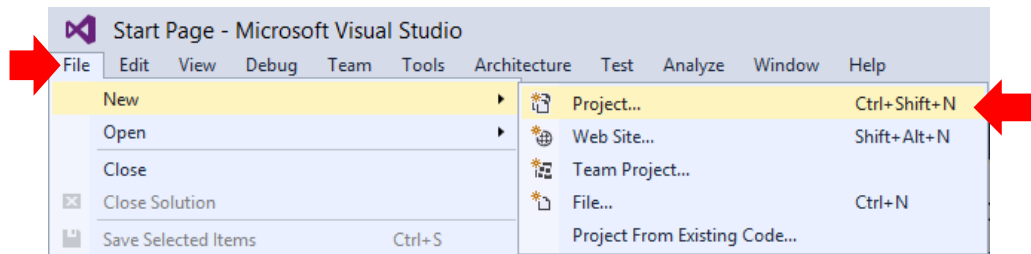
A red arrow points from the text 'Please note that comments here' in the previous block to the first line of the output. A blue bracket groups lines 0 through 2, with a blue arrow pointing to a text box on the right. A red box highlights line 3, with a red arrow pointing to the text 'Try saying "Testing" near the sound sensor' in the next block.

This shows that the environment noise level is around **750 - 1000**

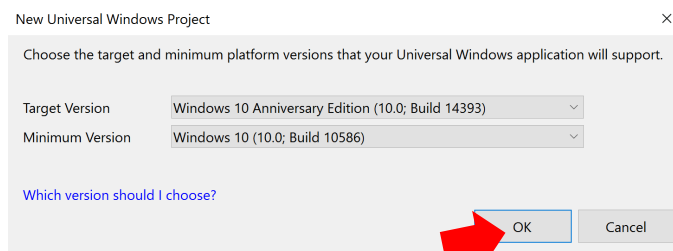
Try saying "Testing" near the sound sensor, you should see a **raise** in the sound value.

Exercise 2: Creating a SoundTrigger

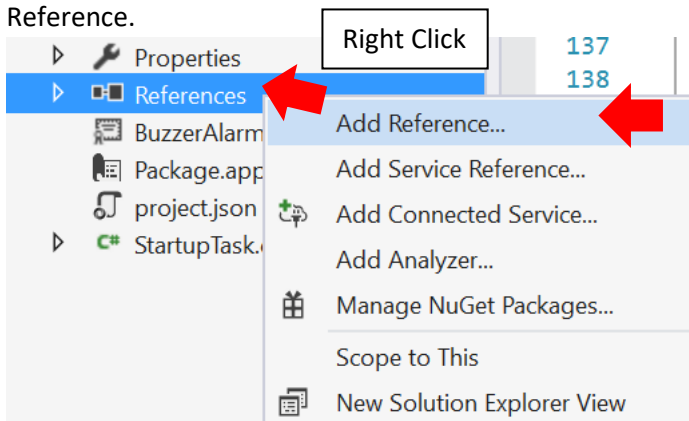
1. In this exercise, we will be creating a program **SoundTrigger**.
2. Similar to previous exercise, use A1 for the sound sensor.
3. Start Visual Studio 2015, Click on File – New - Project.



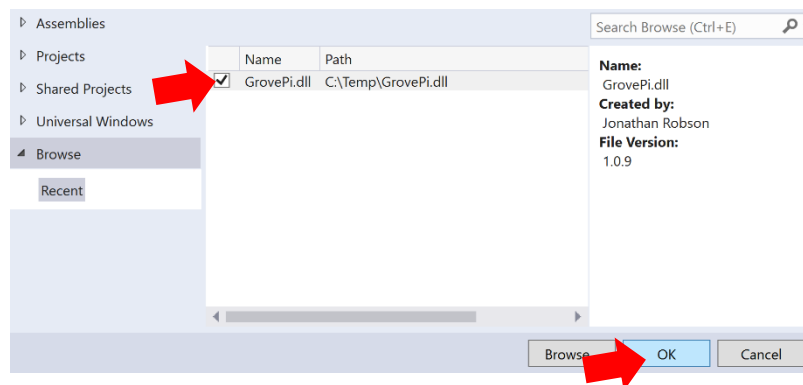
4. From the Installed Templates, select Visual C# – Windows IoT Core – Background Application. Name your project as **SoundTrigger**, save it in you ITP272 folder and Click OK.
5. You should see this pop-up. Click OK.



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



9. Go to your "StartupTask.cs" and type in the following codes above your namespace 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 A0 for sound sensor (mic)
    Pin soundSensor = Pin.AnalogPin0;

    //This is for main logic controller to know that sound has been detected
    private static int sesnsorSound = 0;

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


11. Create the method to get sound values from sensor below the Sleep().

```
//End Sleep()
```

```
private int getSound()
{
    int sum = 0;
    int count = 0;
    int sound = 0;
    while (++count <= 10)
    {
        Sleep(10);
        sound = DeviceFactory.Build.GrovePi().AnalogRead(soundSensor);
        if (sound < 1024)
            sum += sound;
        else
            count--;
    }
    return sum;
} //End of getSound()
```

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

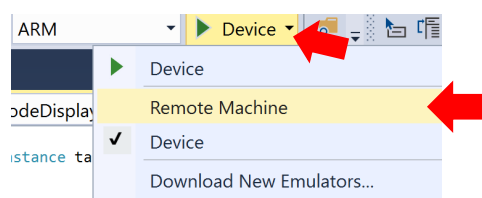
```
public void Run(IBackgroundTaskInstance taskInstance)
{
```

```
    //
    // 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
    //
```

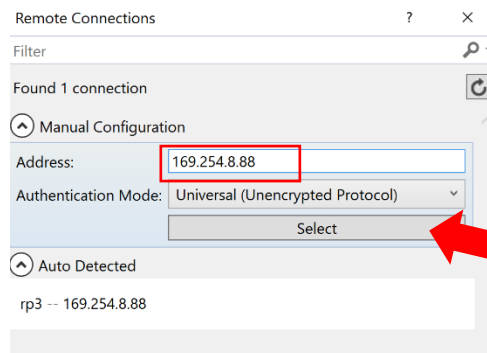
```
    int count = 0;
    Debug.WriteLine("Sound Trigger program started");

    while (true)
    {
        Sleep(300);
        sesnsorSound = getSound();
        if (sesnsorSound > 1200) //Must be bigger than envrionment noise level
        {
            Debug.WriteLine(count++ + " Sound Triggered = " + sesnsorSound);
        }
    }
}
```

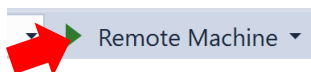
13. Click on the Drop Down button beside the Device and Select **"Remote Machine"** to configure the deployment settings



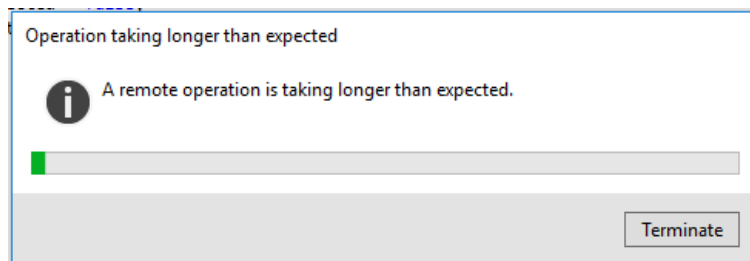
14. Key in the address “**169.254.8.88**” manually as shown below and click on “**Select**” after that. (if you didn’t see this screen, refer to **Practical 1 Exercise 2** on steps to display this screen).



15. Click on Remote Machine to Run the Program.



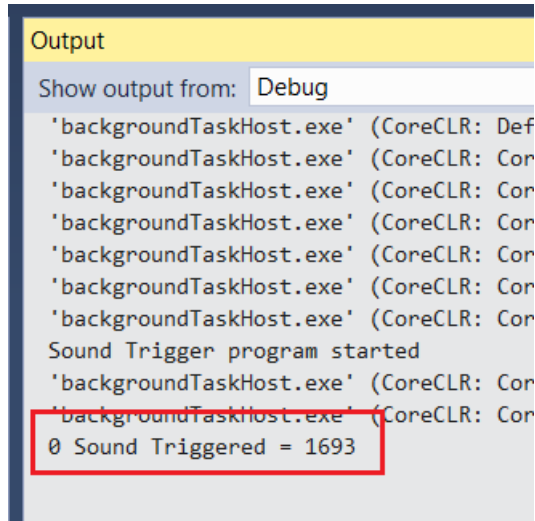
16. You may see the following warning but it is fine. Every time when you deploy a project for the first time on the hardware, it will take a long time to deploy. Just wait for a while and it should be deployed.



17. Once deployed successfully, you can test by going near to the sound sensor and speak (Cannot be too soft and Cannot be too fast)

“Testing Tesing”

The sound sensor should be able to pick up your voice and shows a trigger message at the output window.



```
Output
Show output from: Debug
'backgroundTaskHost.exe' (CoreCLR: Def
'backgroundTaskHost.exe' (CoreCLR: Cor
'backgroundTaskHost.exe' (CoreCLR: Cor
'backgroundTaskHost.exe' (CoreCLR: Cor
'backgroundTaskHost.exe' (CoreCLR: Cor
'backgroundTaskHost.exe' (CoreCLR: Cor
'backgroundTaskHost.exe' (CoreCLR: Cor
Sound Trigger program started
'backgroundTaskHost.exe' (CoreCLR: Cor
'backgroundTaskHost.exe' (CoreCLR: Cor
0 Sound Triggered = 1693
```

Remember, sound sensor may randomly pickup background noise and a different environment has a different background noise. You may need to adjust your threshold level to work in different envrionment.

==End of Practical==