Nanyang Polytechnic

School of Information Technology

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

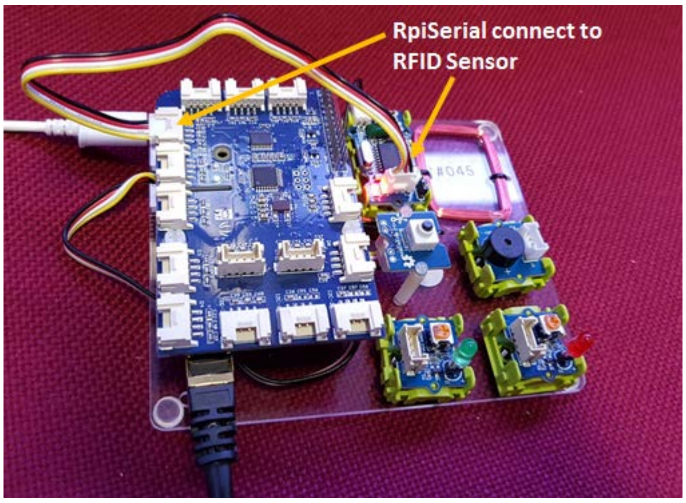
Raspberry Pi Practical : Programming Raspberry Pi with Grove RFID

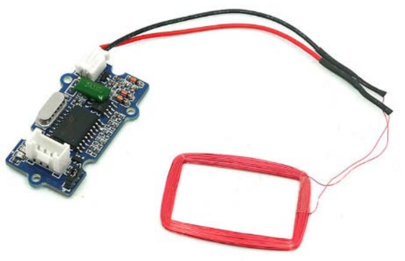
Objectives:

- Learn how to interface with Universal Asynchronous Receiver Transmitter (UART)
- Learn how to create program to interface with the RFID Sensor/Reader.
- Learn how to create program with state machines to work with RFID, buzzer and push button

Exercise 1: Creating RfidDetect

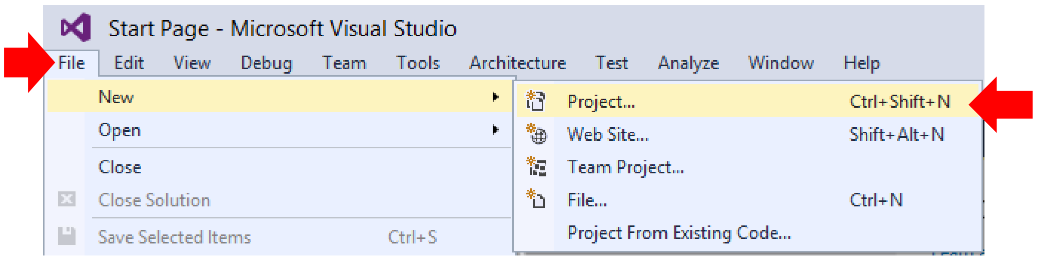
1. Today we will be creating a program that will read a Radio Frequency Identification (RFID) card on the sensor.
2. We shall use UART port **RPISER** of the GrovePi for the RFID Sensor/reader. Ensure that you have port **RPISER** connected to the **RFID Sensor**.



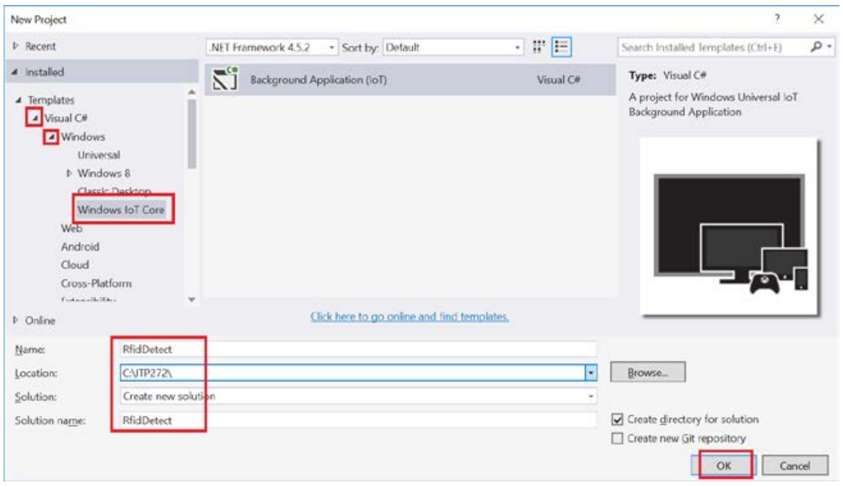


RFID Sensor/Reader

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 **RfidDetect**, save it in you ITP272 folder and Click OK.



Type: Visual C#

A project for Windows Universal IoT Background Application

Name: RfidDetect

Location: C:\ITP272\

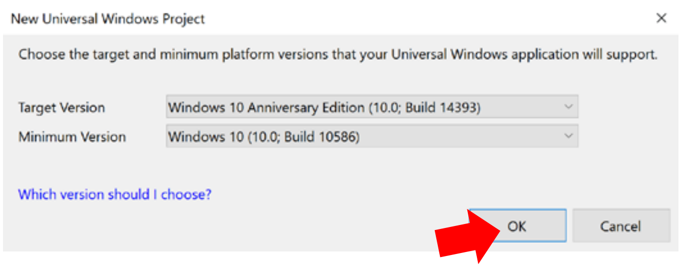
Solution: Create new solution

Solution name: RfidDetect

[✓] Create directory for solution

[ ] Create new Git repository

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



New Universal Windows Project

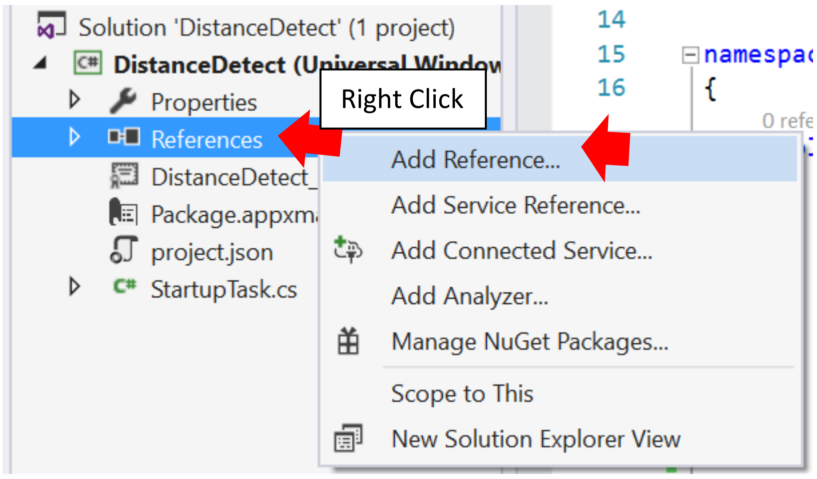
Choose the target and minimum platform versions that your Universal Windows application will support.

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

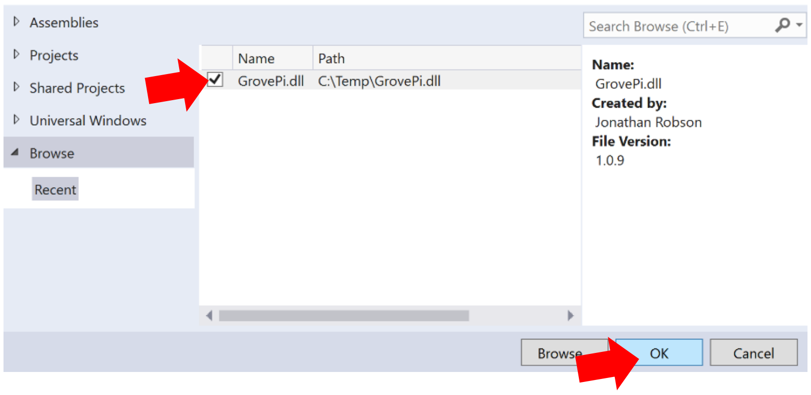
Minimum Version: Windows 10 (10.0; Build 10586)

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.



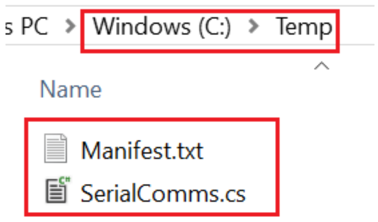
8. Download and select GrovePi.dll as usual. It should appear in your Recent if you've used it before. Check on it and Click OK.



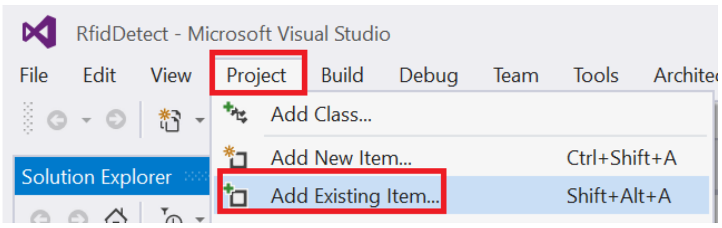
	Name	Path
✓	GrovePi.dll	C:\Temp\GrovePi.dll

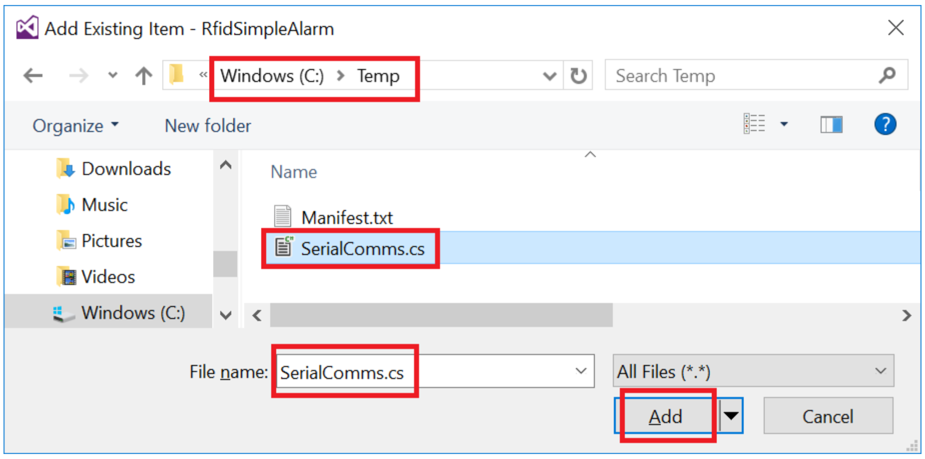
Name: GrovePi.dll
Created by: Jonathan Robson
File Version: 1.0.9

9. Download the starter files into C:\Temp. You should see these files

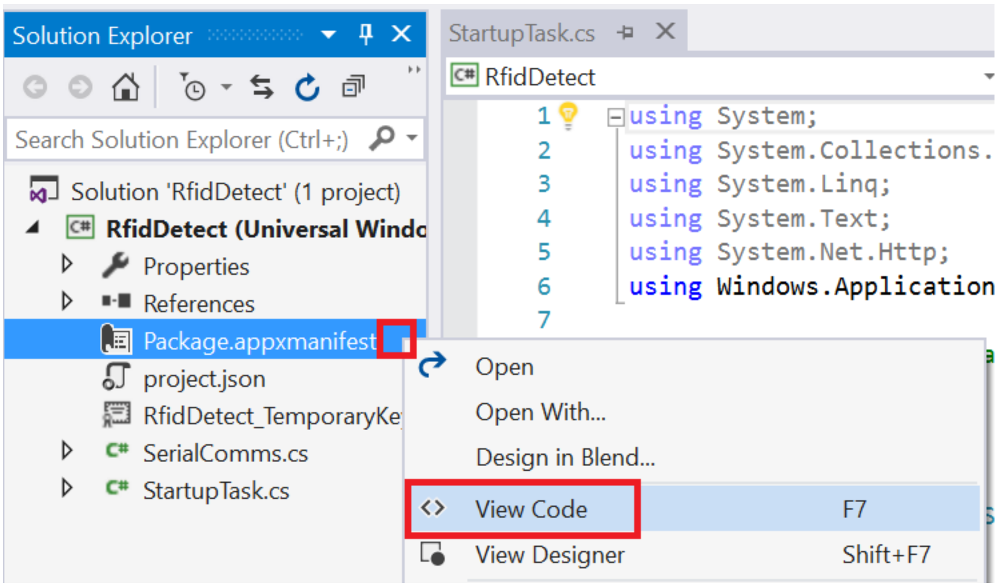


10. Add **SerialComms.cs** to your workspace as below


RfidDetect - Microsoft Visual Studio
File
Edit
View
Project
Build
Debug
Team
Tools
Archite
Solution Explorer
Add Class...
Add New Item... Ctrl+Shift+A
Add Existing Item... Shift+Alt+A

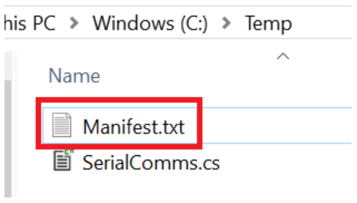


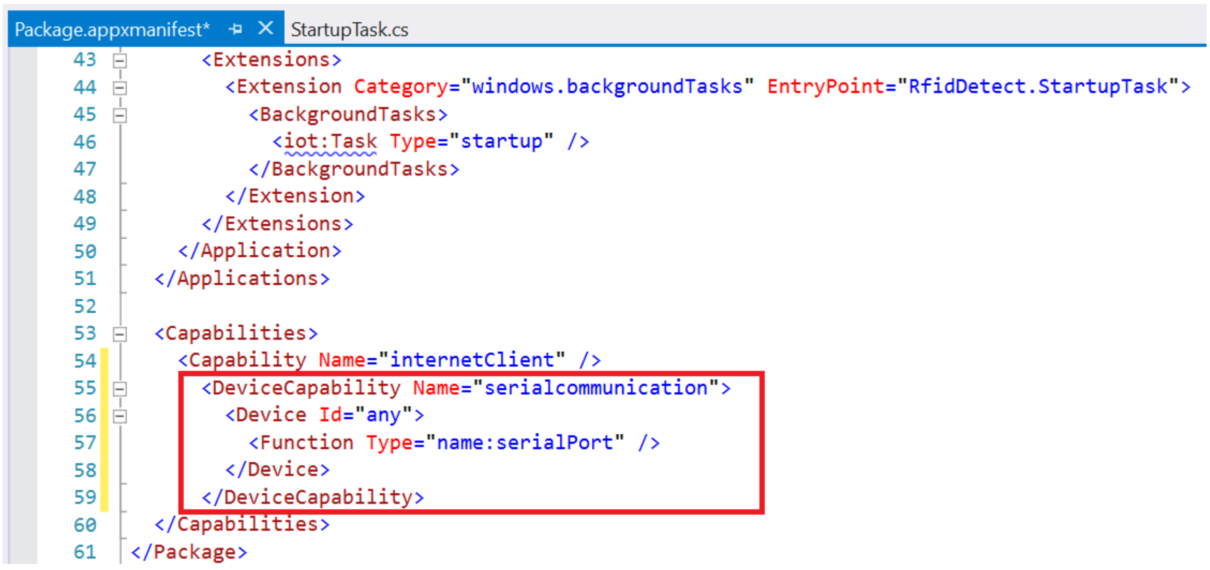
11. **Right click** on the Package.appxmanifest and select View Code as below



```
using System;
using System.Collections.
using System.Linq;
using System.Text;
using System.Net.Http;
using Windows.Application
```

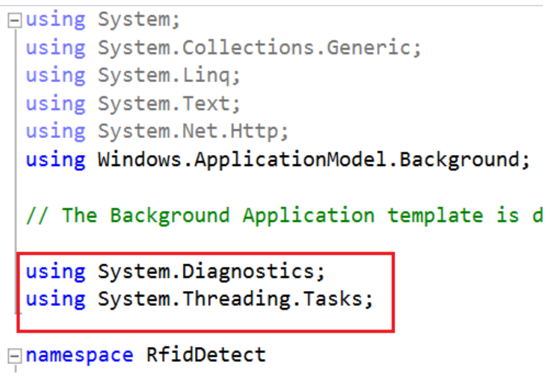
12. Open up the **Manifest.txt** from C:\Temp, copy the settings and paste over to **Package.appxmanifest**





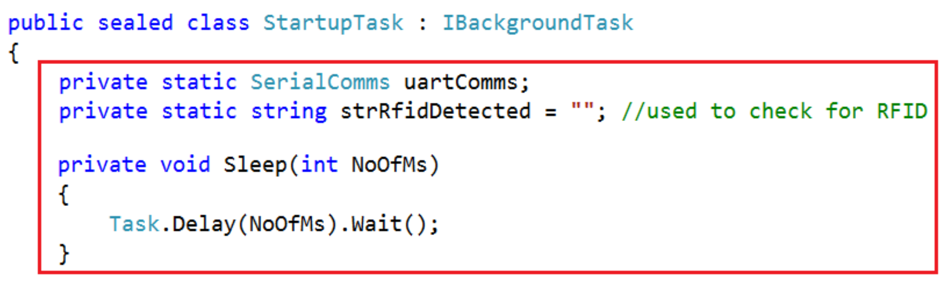
```
      <Extensions>
        <Extension Category="windows.backgroundTasks" EntryPoint="RfidDetect.StartupTask">
          <BackgroundTasks>
            <iot:Task Type="startup" />
          </BackgroundTasks>
        </Extension>
      </Extensions>
    </Application>
  </Applications>

  <Capabilities>
    <Capability Name="internetClient" />
      <DeviceCapability Name="serialcommunication">
        <Device Id="any">
          <Function Type="name:serialPort" />
        </Device>
      </DeviceCapability>
  </Capabilities>
</Package>
```

13. 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.Background;

// The Background Application template is d

using System.Diagnostics;
using System.Threading.Tasks;

namespace RfidDetect
```

14. Add in these codes in the StartupTask.cs



```
public sealed class StartupTask : IBackgroundTask
{
    private static SerialComms uartComms;
    private static string strRfidDetected = ""; //used to check for RFID

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

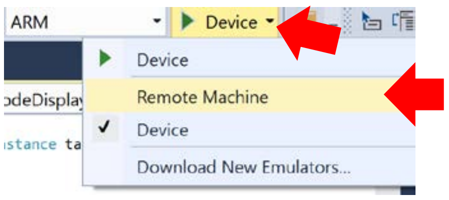
15. Add in the codes below the Sleep(). This is to create the utility methods to handle the Serial comms (uart) data transfer.

```
//this method is automatically called when there is card detected
static void UartDataHandler(object sender, SerialComms.UartEventArgs e)
{
    //strRfidDetected can be used anywhere in the program to check
    //for card detected
    strRfidDetected = e.data;
    Debug.WriteLine("Card detected : " + strRfidDetected);
}

//Must call this to initialise the Serial Comms
private void StartUart()
{
    uartComms = new SerialComms();
    uartComms.UartEvent += new SerialComms.UartEventDelegate(UartDataHandler);
}
```

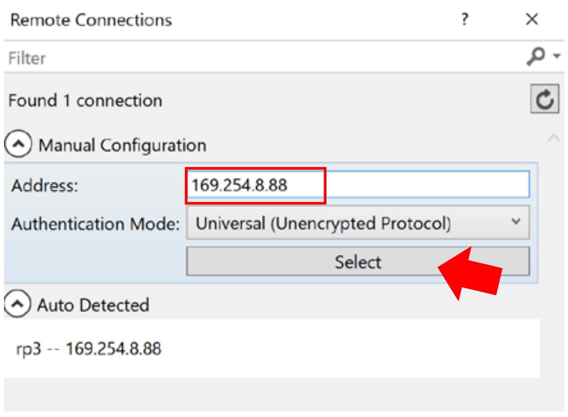
16. Finally, add the codes in the **Run()** method.

```
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

    //Must call this to init the Serial Comm before you can use it
    StartUart();

    while (true)
    {
        Sleep(200);

        if (!strRfidDetected.Equals(""))//this is true for any card detected
        {
            Debug.WriteLine("One Card is detected.");
            Debug.WriteLine("Can you figure out how to check for a specific card?\n");
        }

        //Important: Must always clear after you've processed the data
        strRfidDetected = "";
    }
}
```

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



18. 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

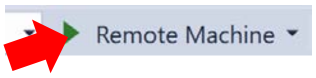
Authentication Mode: Universal (Unencrypted Protocol)

Select

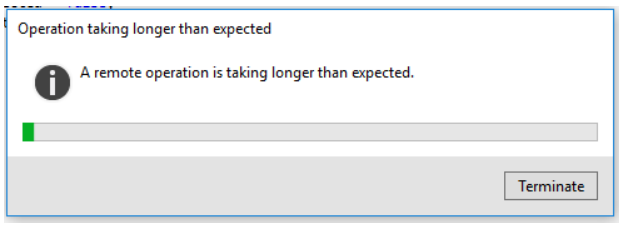
Auto Detected

rp3 -- 169.254.8.88

19. Click on Remote Machine to Run the Program.



20. 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.

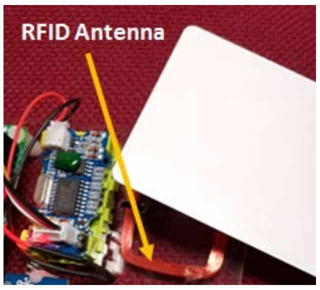


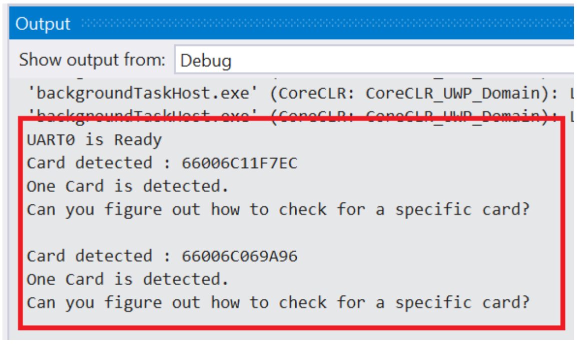
Operation taking longer than expected

A remote operation is taking longer than expected.

Terminate

21. Upon successful deployment, try tapping card on the RFID sensor's antenna. You should see the CARD ID being reported. Tap a different card and see a different ID reported.





Output

Show output from: Debug

```
'backgroundTaskHost.exe' (CoreCLR: CoreCLR_UWP_Domain): L
'backgroundTaskHost.exe' (CoreCLR: CoreCLR_UWP_Domain): L
UART0 is Ready
Card detected : 66006C11F7EC
One Card is detected.
Can you figure out how to check for a specific card?

Card detected : 66006C069A96
One Card is detected.
Can you figure out how to check for a specific card?
```

You've just learnt how to read in RFID Card ID. After knowing your Card Id (let's say it is 66006C11F7EC), you can use the following codes to check for a specific card. You can explore to try it on your cards.

```
if (strRfidDetected.Equals("66006C11F7EC"))
{


}
```

Exercise 2: Creating a RfidSimpleAlarm

Implement a simple alarm system. The system has a panick button used to activate the alarm. Once alarm is activated, you need to use an RFID disarm card to turn off the alarm.

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.

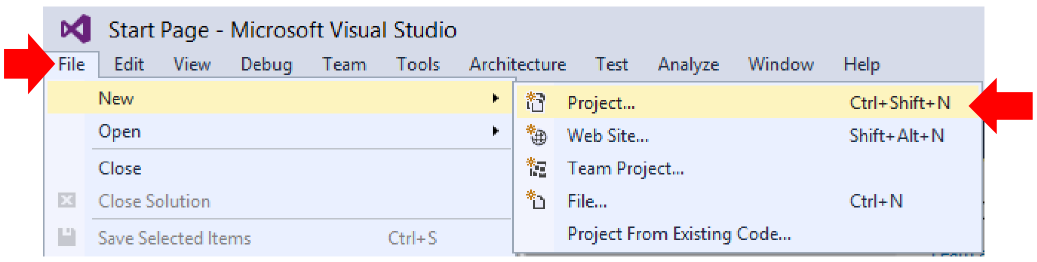
Your program starts in **MODE_Normal**

- **MODE_Normal**
  - Ignore any RFID card tapped on the reader
  - If panic push button is pressed, change mode to **MODE_ALARM**

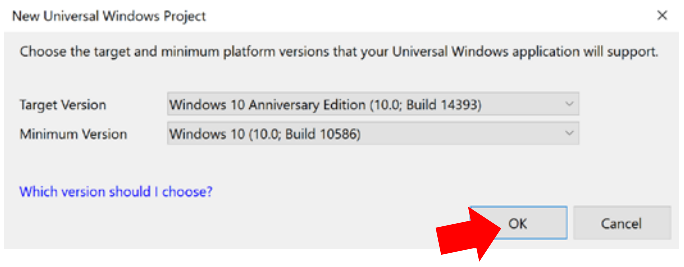
- **MODE_ALARM**
  - Sound the Alarm
  - If any RFID card is detected, change mode to back **MODE_Normal**

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 **RfidSimpleAlarm** as stated in the previous page.
2. Add in Push button and Buzzer. Ensure that you have the following connections
   - RPISER : RFID Sensor/Reader
   - D3 : Buzzer
   - D4 : Push Button
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 **RfidSimpleAlarm**, save it in you ITP272 folder and Click OK.
5. You should see this pop-up. Click OK.

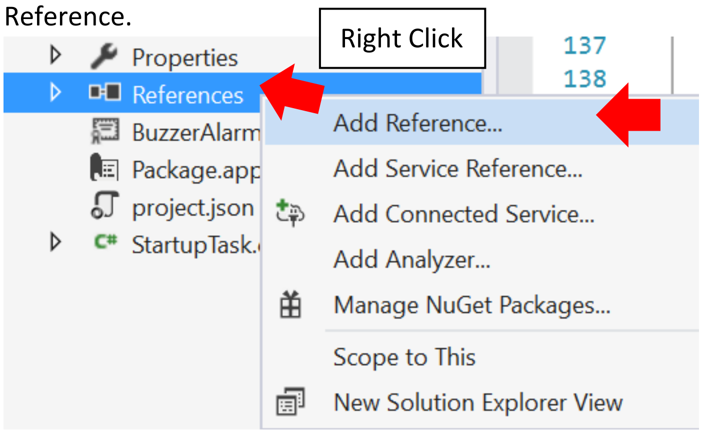

New Universal Windows Project

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

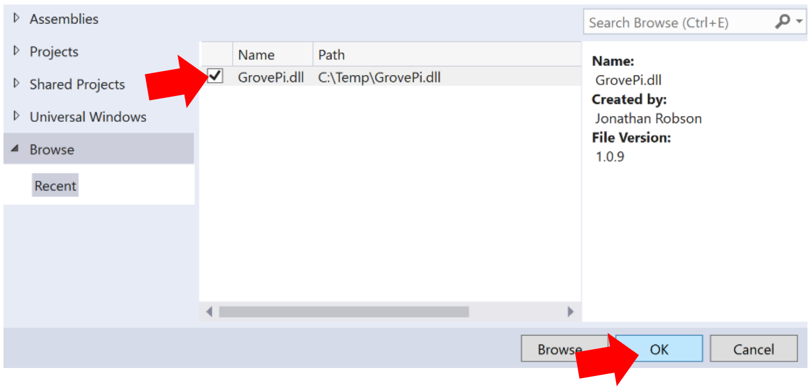
Target Version: Windows 10 Anniversary Edition (10.0; Build 14393)

Minimum Version: Windows 10 (10.0; Build 10586)

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.



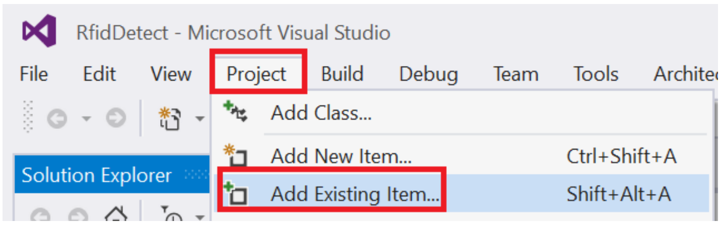
8. Select GrovePi.dll as usual

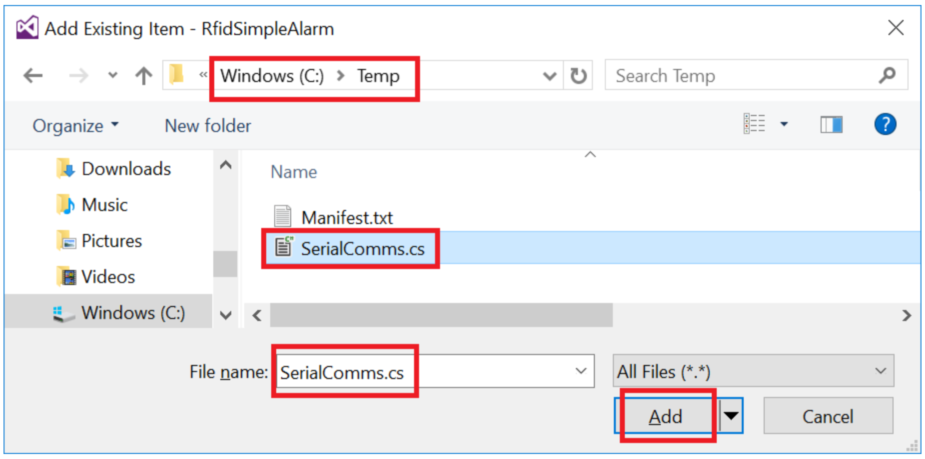


Name	Path
GrovePi.dll	C:\Temp\GrovePi.dll

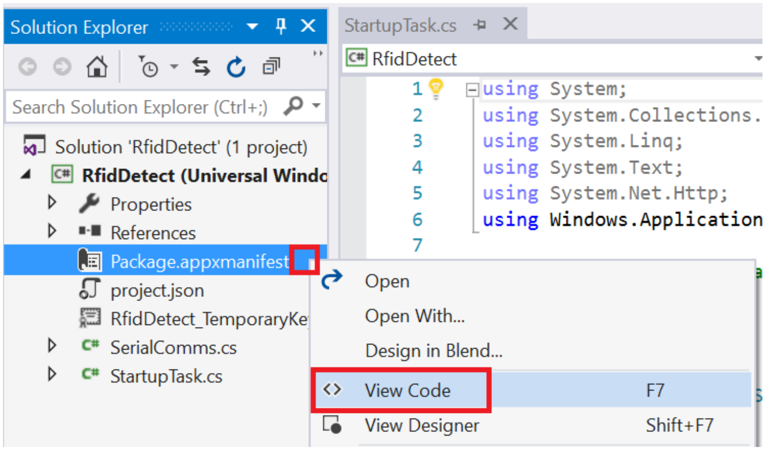
Name:
GrovePi.dll
Created by:
Jonathan Robson
File Version:
1.0.9

22. Add **SerialComms.cs** to your workspace as below


RfidDetect - Microsoft Visual Studio
File
Edit
View
Project
Build
Debug
Team
Tools
Archite
Solution Explorer
Add Class...
Add New Item... Ctrl+Shift+A
Add Existing Item... Shift+Alt+A

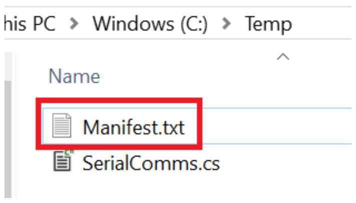


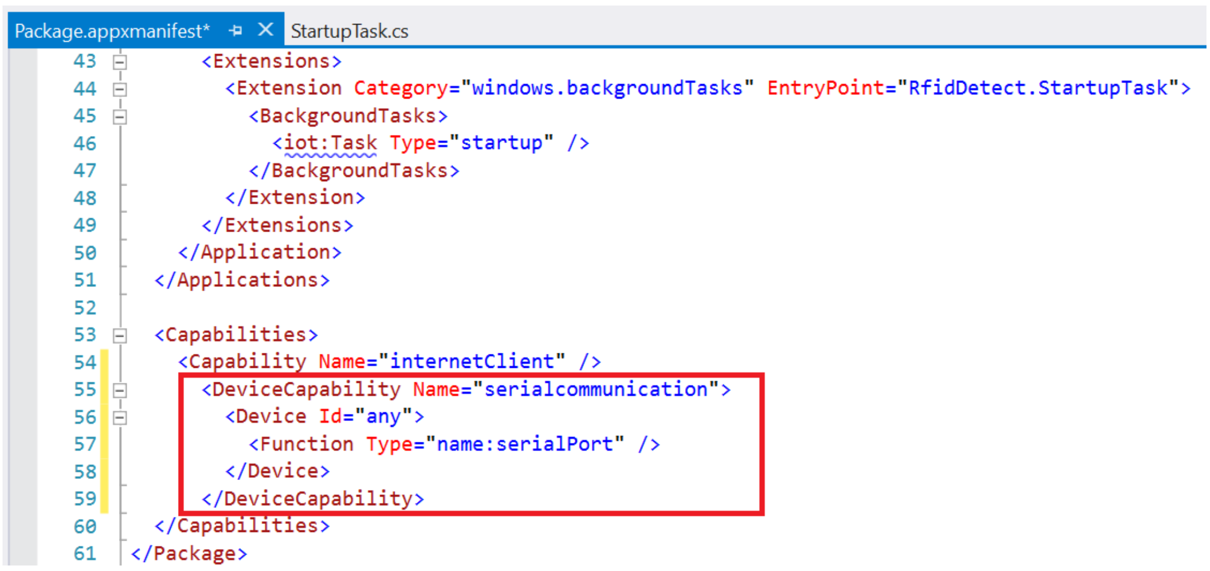
23. **Right click** on the Package.appxmanifest and select View Code as below



```
using System;
using System.Collections.
using System.Linq;
using System.Text;
using System.Net.Http;
using Windows.Application
```

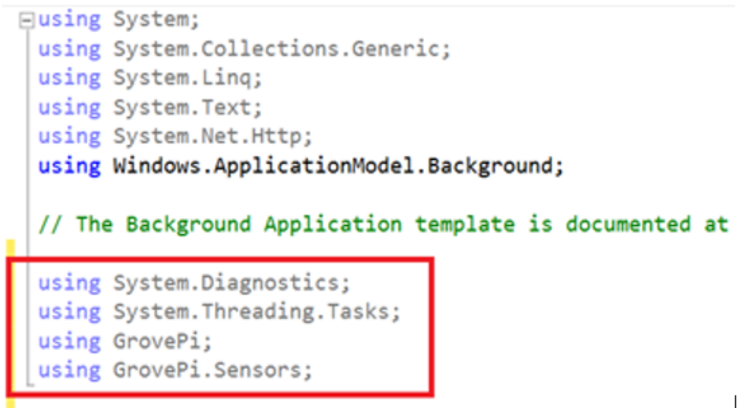
24. Open up the **Manifest.txt** from C:\Temp, copy the settings and paste over to **Package.appxmanifest**





```
      <Extensions>
        <Extension Category="windows.backgroundTasks" EntryPoint="RfidDetect.StartupTask">
          <BackgroundTasks>
            <iot:Task Type="startup" />
          </BackgroundTasks>
        </Extension>
      </Extensions>
    </Application>
  </Applications>

  <Capabilities>
    <Capability Name="internetClient" />
      <DeviceCapability Name="serialcommunication">
        <Device Id="any">
          <Function Type="name:serialPort" />
        </Device>
      </DeviceCapability>
  </Capabilities>
</Package>
```

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_ALARM = 2;
    static int curMode;

    private static SerialComms uartComms;
    private static string strRfidDetected = ""; //used to check for RFID

    //We shall use D3 for Buzzer, D4 for button
    Pin buzzerPin = Pin.DigitalPin3;
    IButtonSensor button = DeviceFactory.Build.ButtonSensor(Pin.DigitalPin4);

    //This is for main logic controller to know that a button is pressed
    private bool buttonPressed = false;

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

11. Create the Serial Comms (UART) methods below the Sleep().

```
}//End of Sleep

//this method is automatically called when there is card detected
static void UartDataHandler(object sender, SerialComms.UartEventArgs e)
{
    //strRfidDetected can be used anywhere in the program to check
    //for card detected
    strRfidDetected = e.data;
    Debug.WriteLine("Card detected : " + strRfidDetected);
}

//Must call this to initialise the Serial Comms
private void StartUart()
{
    uartComms = new SerialComms();
    uartComms.UartEvent += new SerialComms.UartEventDelegate(UartDataHandler);
}//End StartUart()
```

12. Create the method to sound the buzzer

```
}//End StartUart()

private void soundBuzzer()
{
    DeviceFactory.Build.GrovePi().AnalogWrite(buzzerPin, 60);
    Sleep(80);
    DeviceFactory.Build.GrovePi().AnalogWrite(buzzerPin, 120);
    Sleep(80);
    DeviceFactory.Build.GrovePi().AnalogWrite(buzzerPin, 60);
    Sleep(80);
    DeviceFactory.Build.GrovePi().AnalogWrite(buzzerPin, 120);
    Sleep(80);
    DeviceFactory.Build.GrovePi().AnalogWrite(buzzerPin, 0);
    Sleep(2000);
}
```

13. Add the self-monitoring method for the button below the soundBuzzer()

```
}//End of soundBuzzer()
```

```
private async void startButtonMonitoring()
{
    await Task.Delay(100);
    while (true)
    {
        Sleep(100);
        string buttonState = button.CurrentState.ToString();
        if (buttonState.Equals("On"))
        {
            Sleep(100);
            buttonState = button.CurrentState.ToString();
            if (buttonState.Equals("On"))
            {
                buttonPressed = true;
            }
        }
    }
}//End of startButtonMonitoring()
```

14. Create the handler methods for the various states

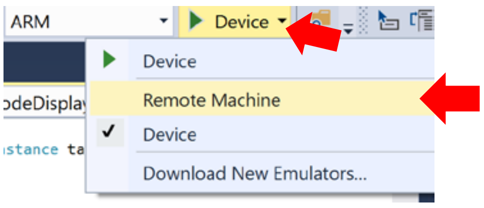
```
}//End of startButtonMonitoring()
```

```
private void handleModeNormal()
{
    //  1.  Define Behaviour in this mode
    strRfidDetected = "";   //ignore and clear any card detected

    // 2.   Must write the condition to move on to other modes
    //button detected to move to Mode ARM
    if (buttonPressed == true)
    {
        buttonPressed = false;

        //Move on to Mode Alarm when button is pressed
        curMode = MODE_ALARM;
        Debug.WriteLine("===Entering  MODE_ALARM===");
    }
}

private void handleModeAlarm()
{
    //  1.  Define Behaviour in this mode
    soundBuzzer();

    // 2.   Must write the condition to move on to other modes
    //card detected to move to Normal
    if (!strRfidDetected.Equals(""))
    //if (strRfidDetected.Equals("66006C13B6AF")) //to detect a specific card
    {
        //Move on to Mode Alarm when button is pressed
        curMode = MODE_NORMAL;
        Debug.WriteLine("===Entering  MODE_NORMAL===");
    }

    //Important: Must always clear after you've processed the data
    strRfidDetected = "";
}
```

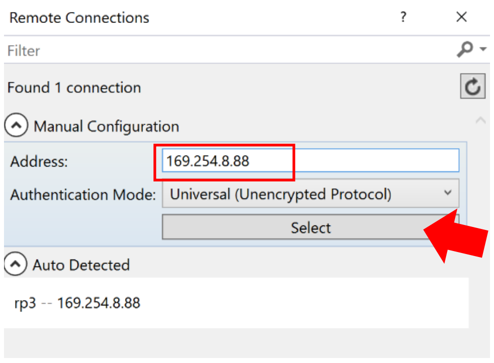
15. Lastly, add the codes for the **Run()** method.

```
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
    //

    //Must call this to init the Serial Comm before you can use it
    StartUart();

    //call button self monitoring method
    startButtonMonitoring();

    //Init Mode
    curMode = MODE_NORMAL;
    Debug.WriteLine("===Entering  MODE_NORMAL===");

     while (true)
     {
         Sleep(300);

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

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



17. 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

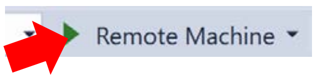
Authentication Mode: Universal (Unencrypted Protocol)

Select

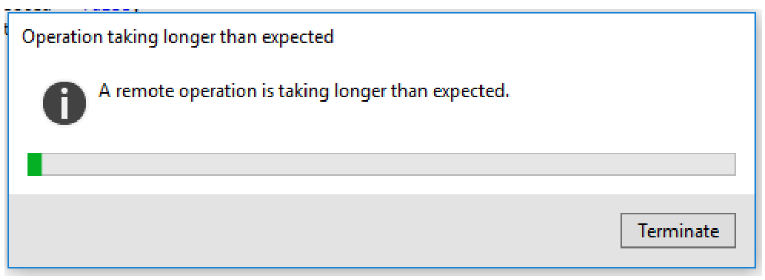
Auto Detected

rp3 -- 169.254.8.88

18. Click on Remote Machine to Run the Program.

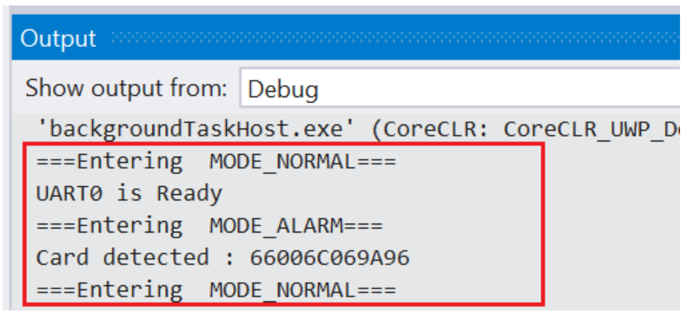


19. You may see the following warning but it is fine. Every time when you deploy a project for the first time on the hardware, it will take a long time to deploy. Just wait for a while and it should be deployed.


Operation taking longer than expected

A remote operation is taking longer than expected.

20. Once deployed successfully, the program will start at MODE_NORMAL. When you press the push button, the application should go to Alarm mode and the buzzer should sound. The alarm will stop sounding when you tap your card.



```
'backgroundTaskHost.exe' (CoreCLR: CoreCLR_UWP_D
===Entering  MODE_NORMAL===
UART0 is Ready
===Entering  MODE_ALARM===
Card detected : 66006C069A96
===Entering  MODE_NORMAL===
```

You can try to modify the program such that only 1 specific card will deactivate the alarm. You've completed the RFID Lab and learnt how to read in RFID from the sensor. Understand the codes in the StartupTask.cs so that you know how to use them and adapt accordingly for your project.

==End of Practical==