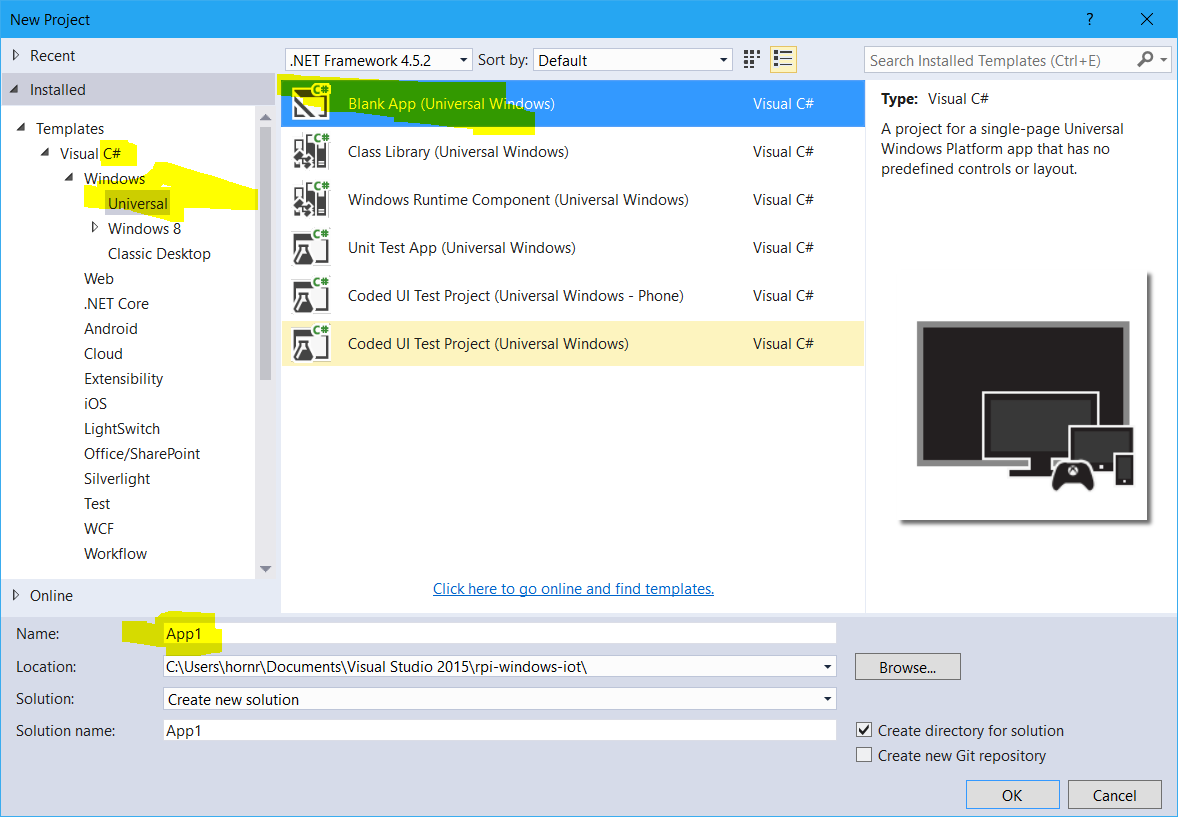
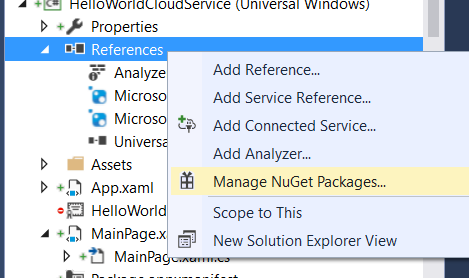
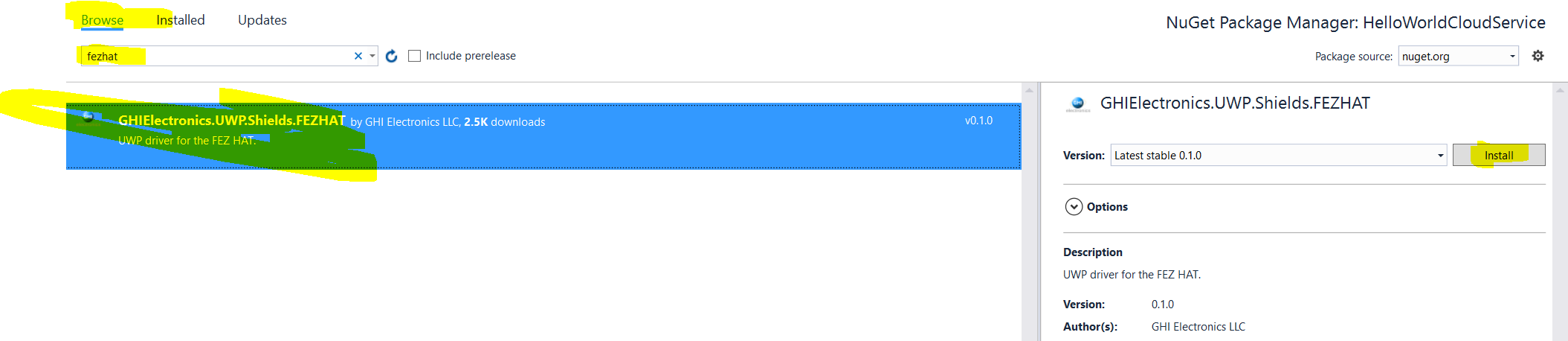
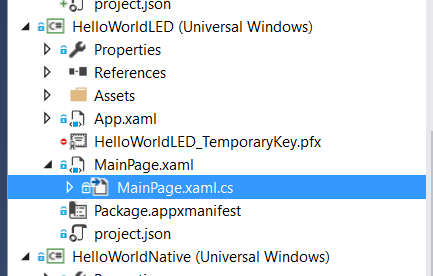
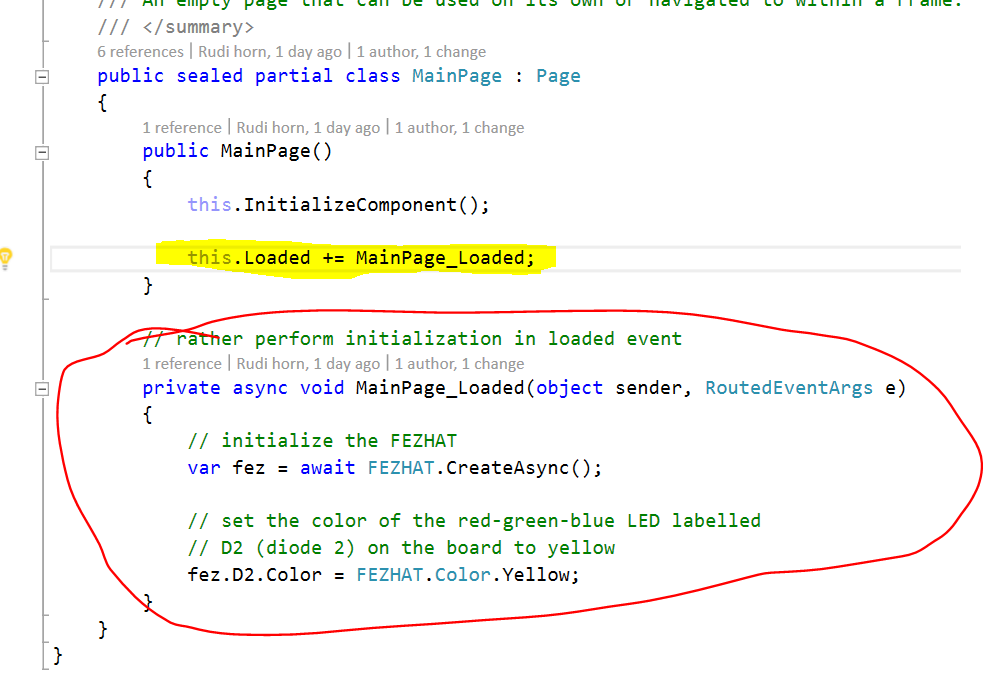
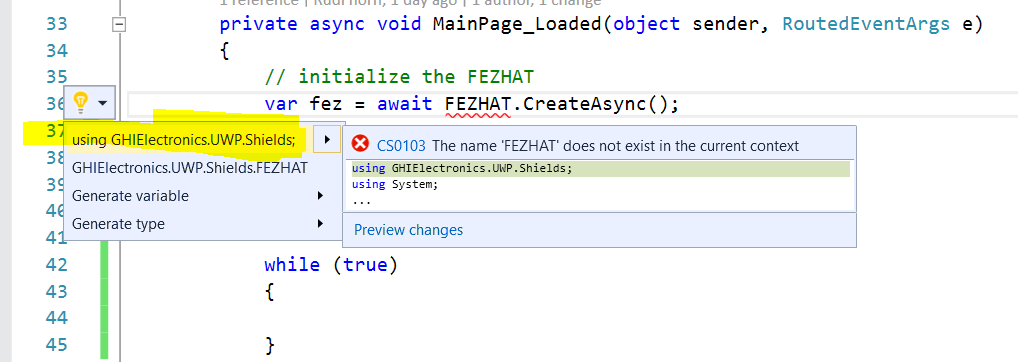
# Raspberry Pi FEZHat Tutorial (Easy)

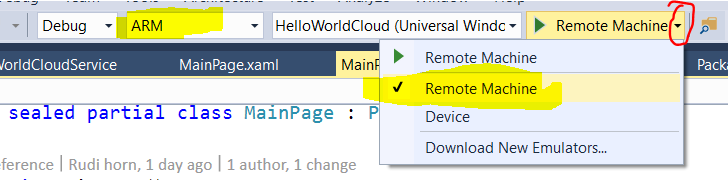
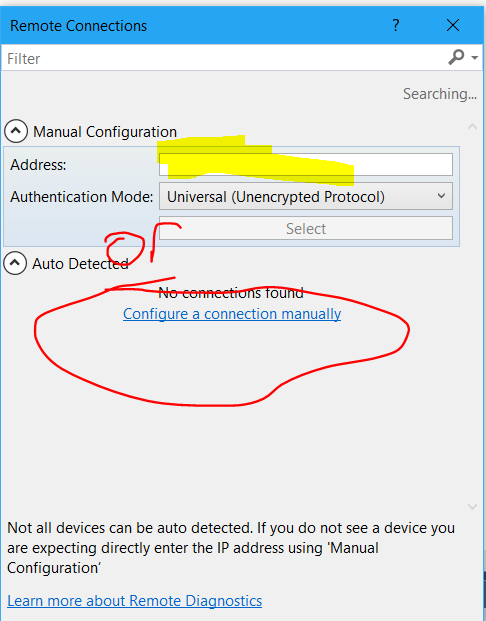
1. Create a new project in Visual Studio (File -> New -> Project, then choose a Universal Windows C# app and choose an appropriate name



1. Open the NuGet package manager  
   
2. Install the FEZHat package:  
   
3. Open the MainPage.xaml.cs file:  
   
4. Extend your code so it looks like the following image  
   TIP: Typing in “this.Loaded+=” in the main method and then pressing the <TAB> key twice will automatically generate the event handler method  
   TIP2: everything behind a // or /// is a comment and you do not need to copy it

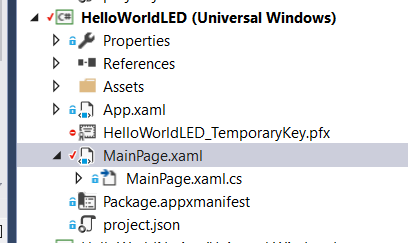
NOTE: you have to manually add the *async* keyword!  
  
If FEZHAT is shown red press <Ctrl>+<.> to show this menu then press the “using GHIELectronics.UWP.Shields” button to add an import statement (this is a line added to the top of your file)  


1. Run your project, the Diode labelled D2 on the FEZHat should now light up yellow ☺

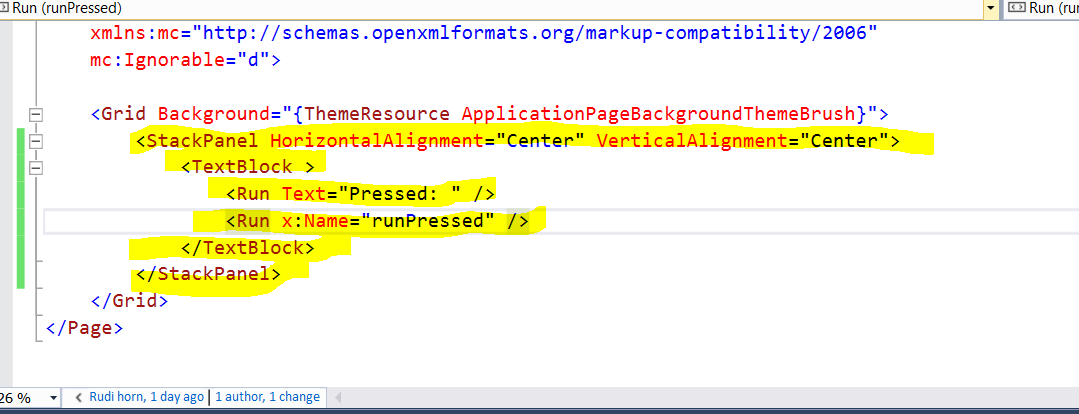
For this you need to make sure it is set to ARM and that remote machine is set to the execution target.  
It should also ask you for the address / target device. If it shows up in auto detection you can just click on it!  
  


## Part 2: Reading button input

1. Switch to MainPage.xaml



1. Add a StackPanel (this holds many graphical elements and just stacks them on top of each other), which is centered in the grid (centering is by setting HorizontalAlignment and VerticalAlignment.  
   Within this StackPanel add a TextBlock (just an element that shows text) which contains two Runs (a run is just a piece of text). The first is given some text which is displayed (“Pressed: “) and the second is given a name, so we can change its Text value in our code!



1. Finally, it is necessary to extend your Loaded method to read the button every 100ms:

NOTE: You will need another import for *Task*TIP: You may think that the “await Task.Delay(100)” call causes your program to stop computing everything else (especially with the infinite while loop!), but actually the *async* keyword causes the rest of the program to be able to continue computation!

