Unknown date | **Unknown author**

Facial Recognition Security System

Summary

Someone at the university asked for advice on a low cost, "smart" security system. He wanted to monitor people entering his dorm room but he didn't want a lot of false alarms since his roommate and friends were often in and out of his room when he wasn't there. The solution - use facial recognition to distinguish between people who are authorized to be there and those who are not. If the person is recognized by the security system, he or she is welcomed into the room; if the person isn't recognized, a photo of the intruder is stored in the system for the room owner to review when he returns.

For this project, I wrote an Universal Windows Platform (UWP) application in C# which runs on Windows 10 IoT Core on a Raspberry Pi 3. I connected a Passive Infrared (PIR) motion sensor to detect when someone enters the room. I included a speech synthesizer and added a speaker to announce that a person has been detected. The attached webcam captures an image of the person and sends it to Microsoft Cognitive Services running on Azure for analysis.

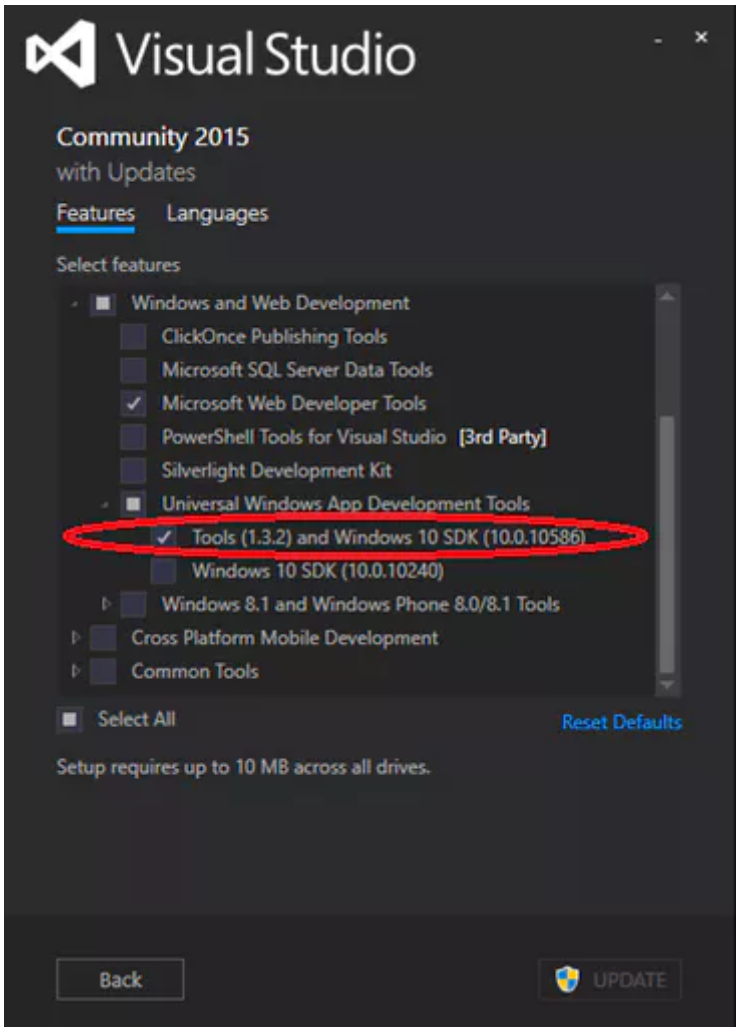
This project was a good introduction to facial recognition and the Microsoft Cognitive Service's Face API made it easier than expected - especially when using their open source SDK. I also resused a lot of the Facial Recognition Door Lock project created by the Windows IoT Team.

Prerequisites

- You'll need to have Windows 10 (version 10.0.10240 or better) running on your development machine.
- You'll need the Windows 10 IoT Core Dashboard on your development machine.
- You'll need to image an SD card with Windows 10 IoT Core - minimum supported OS version for this project is build 10.0.14295.
- You'll need Visual Studio 2015 Update 2.

Installing all the prerequisites could take up to an hour or so but most of that time is unattended while the bits download and get installed so plan accordingly. Microsoft has documented the setup process well in a step-by-step guide.

With respect to Visual Studio 2015 Update 2, you may need to customize your installation (if you are installing it for the first time) or modify your existing installation to ensure that you have the most recent version of the Window 10 SDK installed - the Windows 10 SDK is not installed by default in the the Community Edition and it is required for this project (I believe it is installed by default in the Professional and Enterprise editions).



Update your installation of Visual Studio 2015 Update 2 to ensure that the Windows 10 SDK is installed.

To verify that you completed the setup process correctly use the Windows 10 IoT Core Dashboard; the Dashboard will discover the Pi once Windows 10 IoT Core has booted and connected to the network (you can plug a network cable between the Pi and your network's router or directly into your development machine's Ethernet port if you turn on Internet connection sharing).

Use the device portal to configure WiFi. To open the device portal, go to the 'My Devices' tab in the Windows 10 IoT Core Dashboard and click on the globe under the 'Open in Device Portal column'. You'll be prompted to enter the credentials for the device - "administrator" and "p@ssw0rd" by default. Go the Networking page to setup WiFi.

Components

The following components are required for this project. In the United States, total component costs are less than $75.

- Raspberry Pi 3 (this project works equally well on the Raspberry Pi 2 but you would then need a separate WiFi dongle if you want to go wireless) (you could also use an Intel MinnowBoard Max or a Qualcomm DragonBoard 410c).
- Passive Infrared Motion Sensor
- Supported webcam (I used the Microsoft LifeCam HD-3000 but others are support as well; check the list here)
- Speaker (plug the speaker's 3.5mm audio jack directly into theRaspberry Pi; I'd recommend powering the speak separately)
- Jumper wires

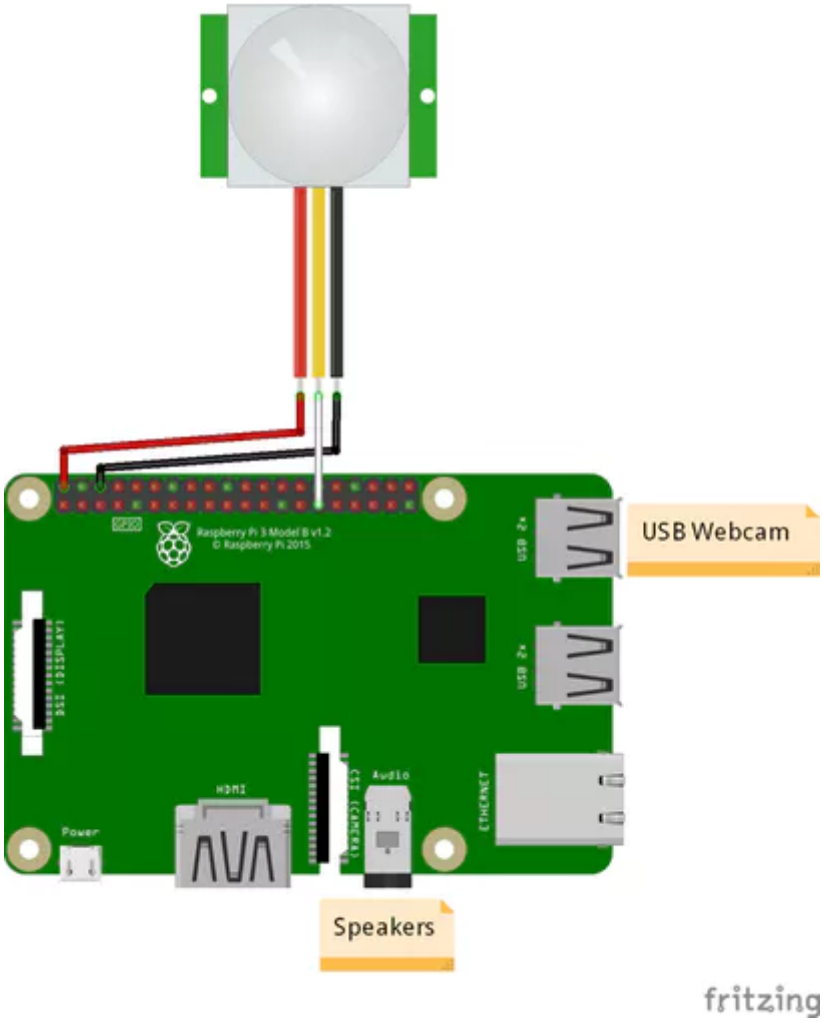
Instructions - Follow these steps

Step 1 - Wire Up the Security System

Time - 5 minutes

The security system is comprised of a PIR sensor, speaker, and webcam connected to a Raspberry Pi 3. The PIR sensor will detect when someone enters your room. The speaker plays the text to speech messages. And, the webcam will capture an image which will be processed by the facial recognition service.

First, connect the PIR motion sensor's pins to the Raspberry Pi using jumper wires - 5V on the sensor to 5V PWR on the Pi (I used pin 2), GND on the sensor to GND on the Pi (I used pin 6), and signal on the sensor to a GPIO pin on the Pi (I used pin 29 / GPIO 5). Double check the data sheet for your sensor to make sure you correctly identify the sensor's 5V, GND, and signal pin as their order varies by manufacturer. (I bent the pins on a female header so that I could mount the sensor upright on the mini breadboard.)



1 / 2 · Plug in your webcam and speakers and wire up the PIR sensor.

My PIR sensor operates at 5 volts which is why it's power pin needs to be connected to a 5V power pin on the Pi. But, it's signal pin operates at only 3.3V which is why it can be connected directly to a GPIO pin. Again, double check the data sheet for your sensor to ensure that the signal's voltage is 3.3V. If it's 5V, you'll have to divide the voltage before connecting it to the Raspberry Pi's GPIO pin.

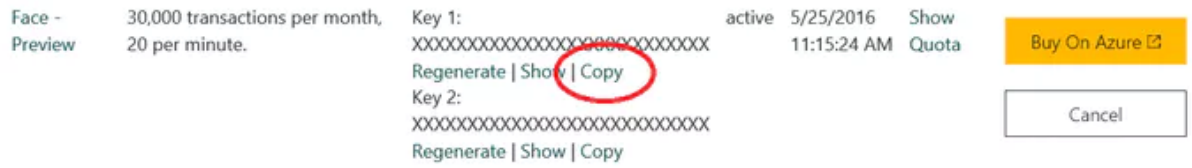
There is jumper on the back of the PIR sensor that lets you select "Single Trigger" or "Repeat Trigger". Ensure that the jumper is set to Single Trigger so that once a person enters your room, he/she doesn't have to put his/her face in front of the camera every few seconds to be recognized.

Next, connect the webcam to one of the Raspberry Pi's USB ports. And, connect your speaker to the Raspberry Pi's audio jack.

Step 2 - Sign Up for Microsoft Cognitive Services

Time - 5 minutes

In order to use Microsoft Cognitive Services, you must have a subscription. There is a free tier for the Face API which is more than sufficient for this project (the free tier allows 30k transaction per month). Sign up here. Make sure you check Face API when setting up your subscription. Once you have setup your subscription, copy the key for the Face API - you will need it in the next step.



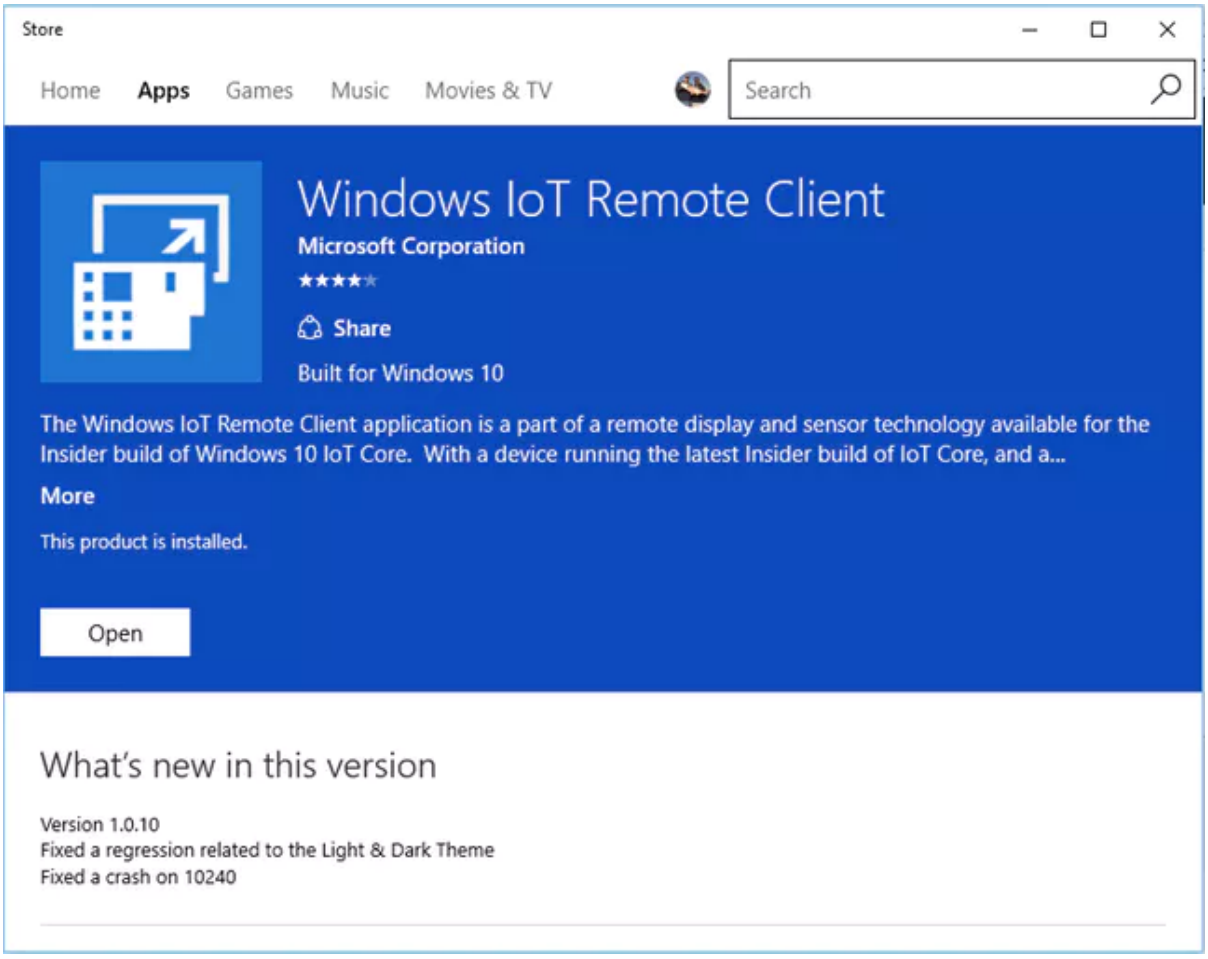
Copy your subscription key for the Face API as you'll need to add it to the app.

Step 3 - Add a Display, Keyboard, and Mouse

Time - 5 minutes

This project requires a display as you'll need to interact with the UI to setup facial recognition and to review photos of intruders. If you have an external HDMI monitor, keyboard, and mouse, connect those now and move on to step 4. If not (I only have laptops and don't have any separate displays), you can use the remote display feature.

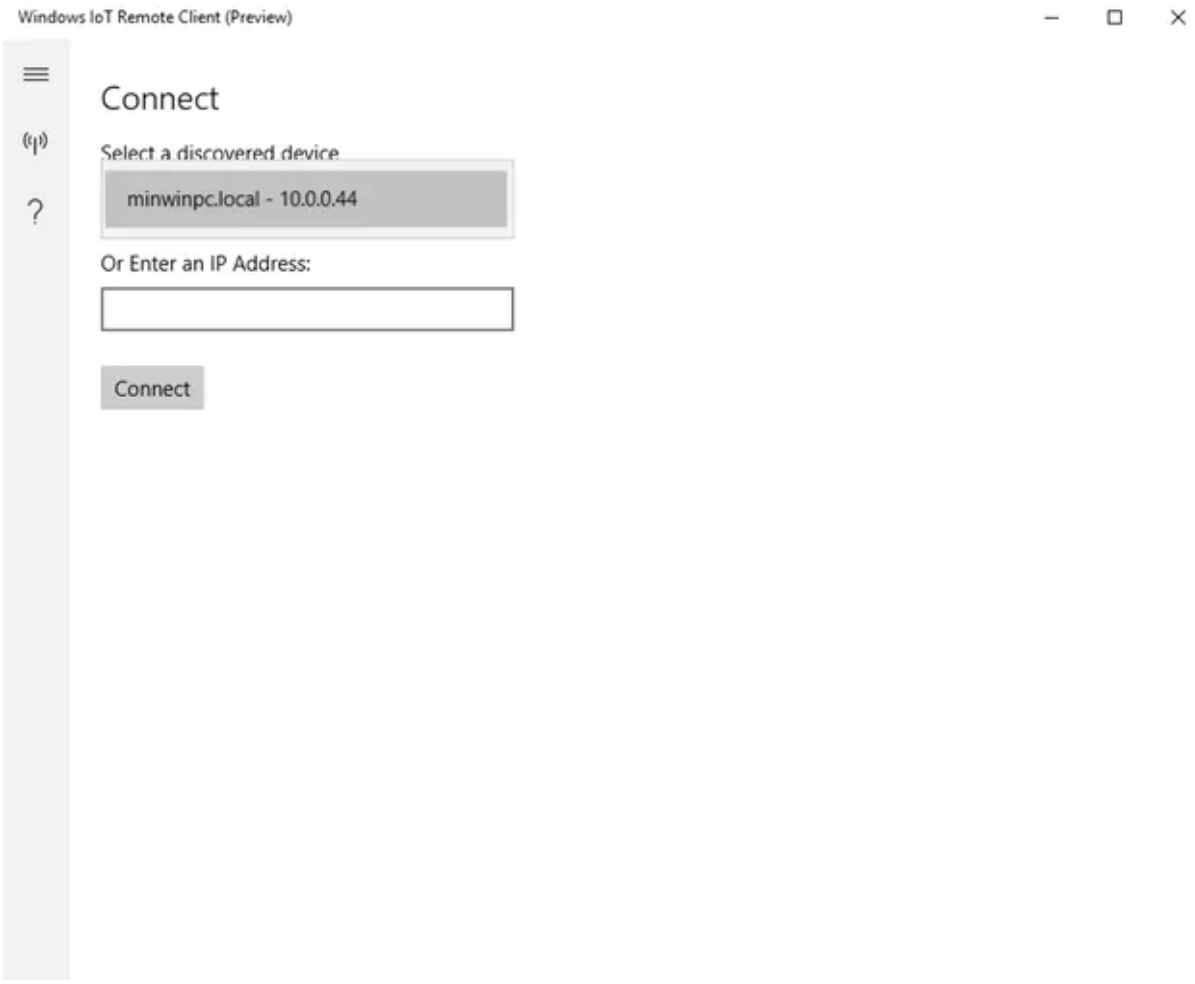
The remote display allows you to remotely control your UWP app running on Windows 10 IoT Core. You can use the display from any of your other Windows 10 devices (phone, tablet, laptop, desktop) to view the app's UI and the keyboard, mouse, and touch of your Windows 10 device to interact with and to provide input to the app. You'll just need to get the remote display app from the Store and install it on your Windows 10 device - here is the link to the Store. The only limitation is that the Windows 10 device need to be on the same network as your Raspberry Pi running Windows 10 IoT Core.



1 / 2 • Download the Windows IoT Remote Client app from the Store onto your dev machine.

Once you have install the remote display client on your Windows 10 device - i.e. your development machine - you'll need to log in to the Raspberry Pi and enable the remote server. Open the Device Portal in your browser: navigate directly to the Raspberry Pi (http://{Pi's IP address}:8080) or open Windows 10 IoT Core Dashboard, go to the 'My Devices' tab, and click on the globe under the 'Open in Device Portal' column. Enter the credentials for your Raspberry Pi and then click on the 'Remote' menu option. On the Remote Server page, click the check box to enable the Windows IoT Remote Server.

Now, launch the remote display client app, select your Raspberry Pi from the drop down, and click the 'Connect' button. The display from your Raspberry Pi will be shown. You'll notice that you can now use the mouse from the machine running the remote display client to interact with the Raspberry Pi.



1 / 2 • Launch the remote display client and select your Raspberry Pi from the drop down.

Step 4 - Deploy the App to the Raspberry Pi

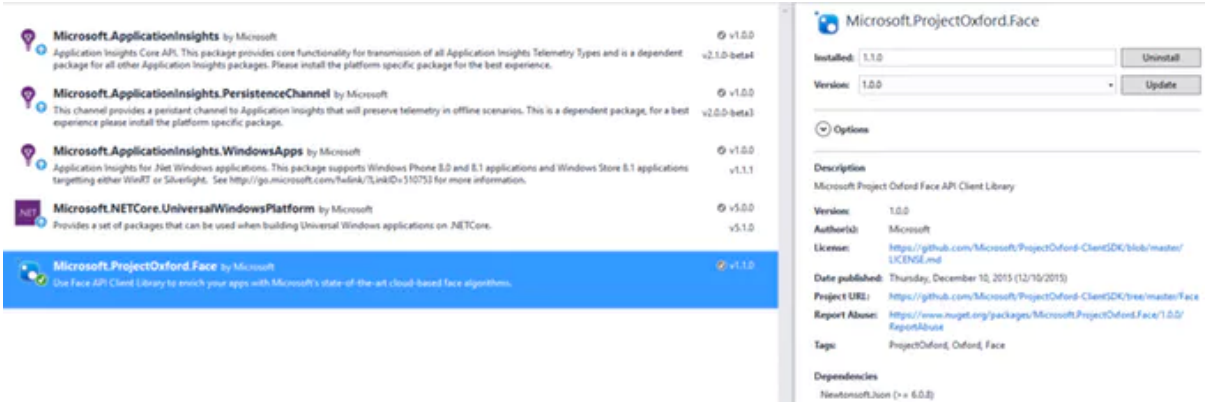
Time - 10 minutes

Download the code from the GitHub repository and open the solution in Visual Studio 2015. Before building the project, you will need to update the GeneralConstants class with your Face API subscription key. (Microsoft Cognitive Services was formerly known as Project Oxford so you will see several references to Oxford - especially in the client libraries.)

```
public const string OxfordAPIKey = "";
```

The solution should automatically pull in the all references required for this project including the Microsoft.ProjectOxford.Face NuGet package. The

project was built using version 1.1.0.



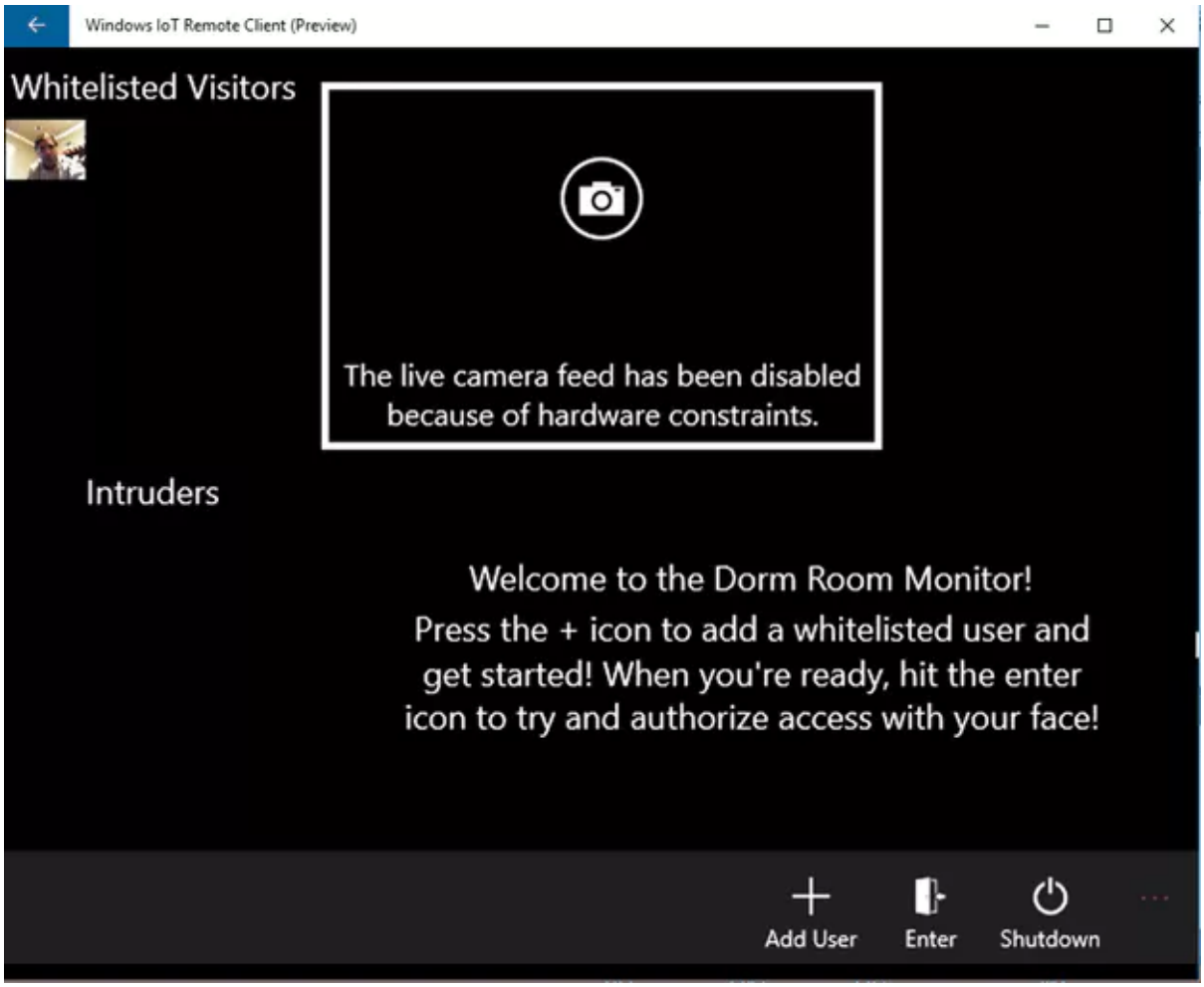
Now, build the project and deploy the app to the Raspberry Pi.

Step 5 - Create Your Whitelist

Time - 5 minutes

When someone enters your room, the PIR sensor detects their presence and takes a picture with the webcam. The app then uses Microsoft Cognitive services to detect the face in the picture and to attempt to recognize it. The face in the picture is compared against the list of people approved to be in your room - i.e. the whitelist. You must first define the whitelist before anyone can be recognized.

From the app's main screen, click the '+' icon at the bottom to add someone to the whitelist. The next screen will allow you to take a picture of the person and enter his/her name. Then, click 'Confirm'. Now, that person has been added to the whitelist.



1 / 4 • Click the 'Add User' button to add a new person to the whitelist.

However, to improve recognition, it's recommended that you have additional images for each person. So, from the main screen, click on the person's face in the 'Whitelisted Visitors' section. Now, click the 'Add Photo' button to capture additional images of the person's face. When finished, clicked the 'Go Home' button to return to the main screen. You can also use the 'Delete User' button whenever you want to remove someone from the whitelist entirely.

Once you've added everyone who should have access to your room to the whitelist, you are ready to start monitoring. The main screen has an 'Intruders' section, This is where you will see photos of anyone who entered your room and who wasn't recognized. If you want to access the images separate from the app or what to copy them to some other location, you can find them on the SD card at this location:

```
C:\Data\Users\DefaultAccount\Pictures\Dorm Room Monitor
Intruders
```

Platforms

This app can also be run on your development machine (or Windows phone) directly - it really is an Universal Windows Platform app. (I found this helpful as I started to develop the app on my laptop while I was waiting for the Raspberry Pi 3 to arrive.) If you run it on a non-Windows IoT device, you will not have GPIO pins with which to connect to the PIR motion sensor. Therefore, there is a button on the main screen's UI to "trigger" the motion sensor - just click on the 'Enter' button.

In addition to the Raspberry Pi 2 and 3, Windows 10 IoT Core is supported on the Intel MinnowBoard Max and the Qualcomm DragonBoard 410c. Both the MinnowBoard Max and the DragonBoard 410c have GPU support in Windows 10 IoT Core. Therefore, the app UI can display the live stream from the webcam. To enable live video on either of these devices, change the DisableLiveCameraFeed variable to 'false' in Constants.cs:

```
public const bool DisableLiveCameraFeed = true;
```
