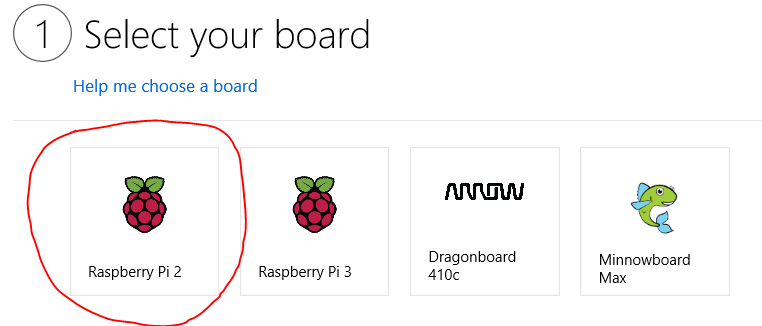
# Download and install the Windows IoT Core Dashboard

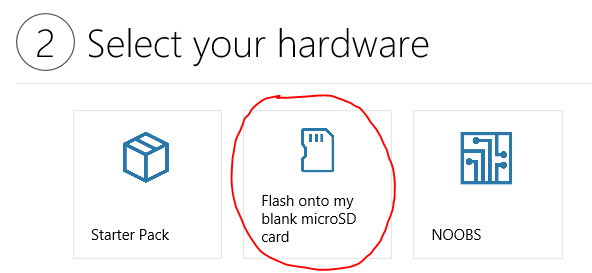
1. Insert the microSD card provided with the car kit into a card reader. If do not have access to a microSD card reader, then use the microSD adapter provided with the card.
2. Navigate to [www.windowsondevices.com](http://www.windowsondevices.com)
3. Click “Get started now” as shown below.



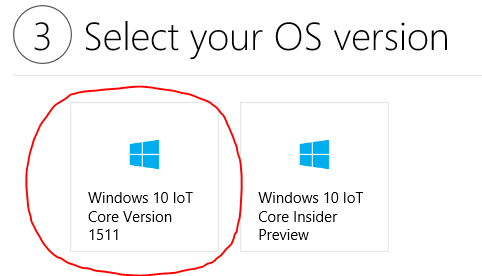
1. On the following page – select Raspberry Pi 2 as shown



1. Then select – Flash onto my blank microSD card.



1. Then select – Windows 10 IoT Core Version 1511



1. Click “Next”.
2. From the next page - Download and install the Windows IoT Core Dashboard.

# Flashing Windows 10 on Raspberry Pi

1. **Open the Windows 10 IoT Core Dashboard** you downloaded and installed from the [Get the tools](https://developer.microsoft.com/en-us/windows/iot/win10/GetStarted/rpi2/sdcard/stable/getstartedstep1) page.
2. **Click "Set up a new device".**
3. **Select Raspberry Pi 2 from the dropdown.**

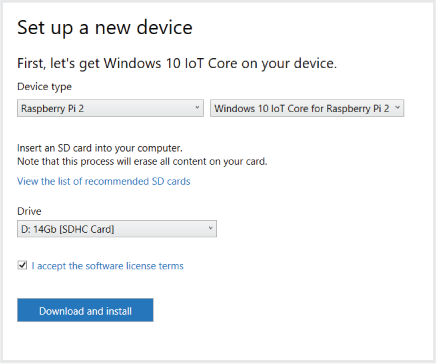
Each board has its own specialized image.

1. **Download and install Windows 10 IoT Core on your board.**

A window will pop up to show you the progress. This step can take several minutes depending on the speed of your SD card.

1. **Plug the SD card into your board and power on.**

You have the option to plug in a display, but it is not necessary. For best results, plug it in before you power on the device. First boot will take several minutes as the operating system does its initial installation.



Insert the microSD card in the back of the Raspberry Pi as shown below

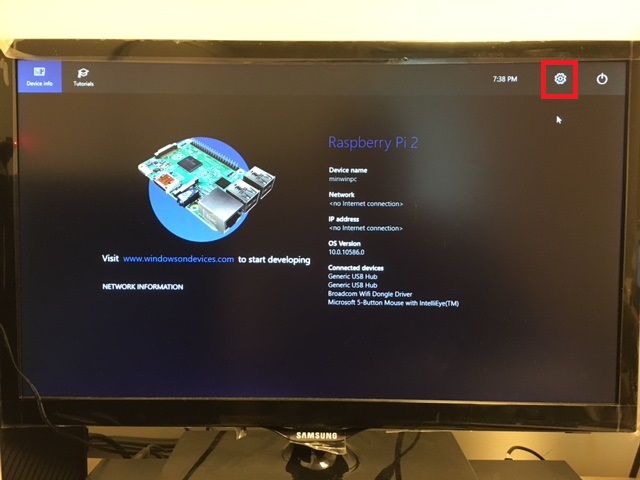


Insert the white colored WiFi dongle provided in the car kit into any usb slot on the board. Insert a micro USB to USB cable in the bottom left corner of the Pi. This will power the Pi and boot to Windows 10. First boot will take several minutes as the operating system does its initial installation. **Optionally**: you can also have a HDMI display, a usb keyboard and mouse connected to the board before powering it up for the first time.

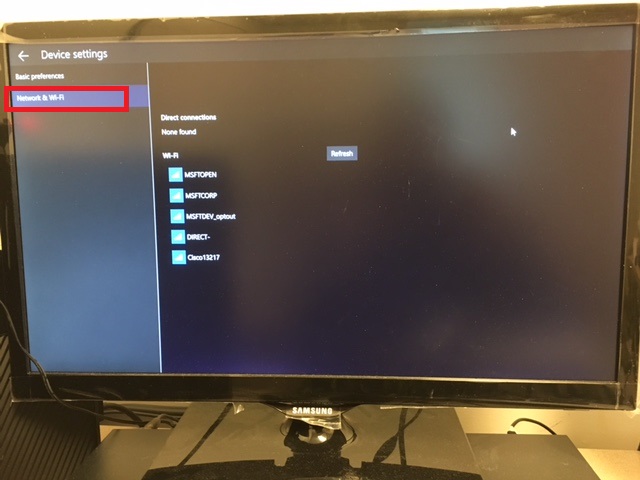


# Connect the board to the network

1. If you have a HDMI display, keyboard and mouse connected to the Raspberry Pi-
   1. Click on the Settings icon (gear) on the top right corner of the screen



* 1. Select Network & WiFi on the left panel



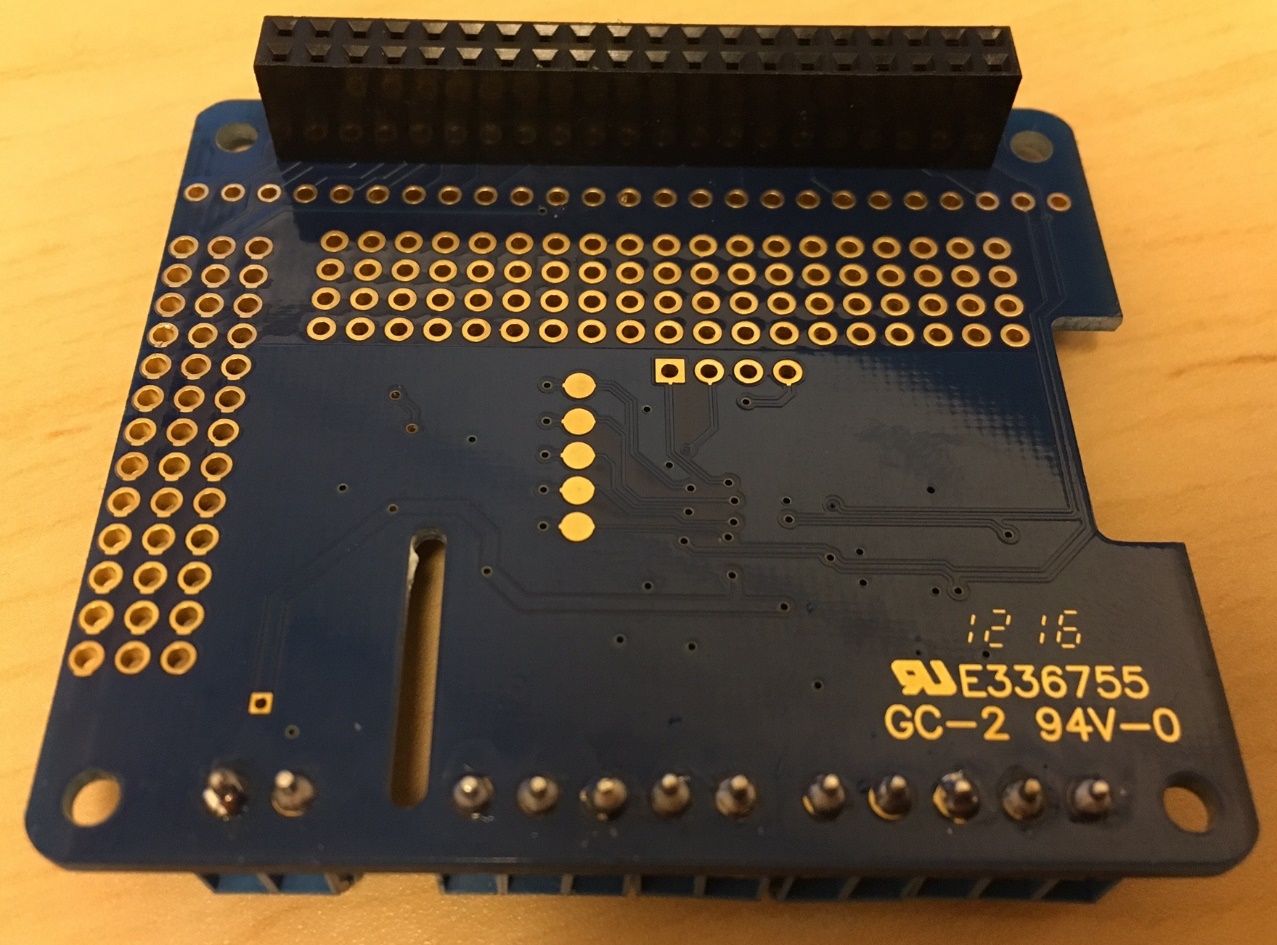
* 1. Select the network you want Raspberry Pi to join, and enter the password.

1. Alternative -
   1. In the Windows 10 IoT Core Dashboard - Go to "My devices".
   2. Find your board and click "Configure Device". If your board has a WiFi adapater and it has not yet been set up, it will start to broadcast itself as a network. Unconfigured boards will begin with "AJ\_" (e.g. AJ\_58EA6C68). If you don’t see your board, make sure that you’ve allowed enough time for your board to boot. If all else fails, reboot your device.
   3. Enter your network credentials. Your computer will now connect to your board.

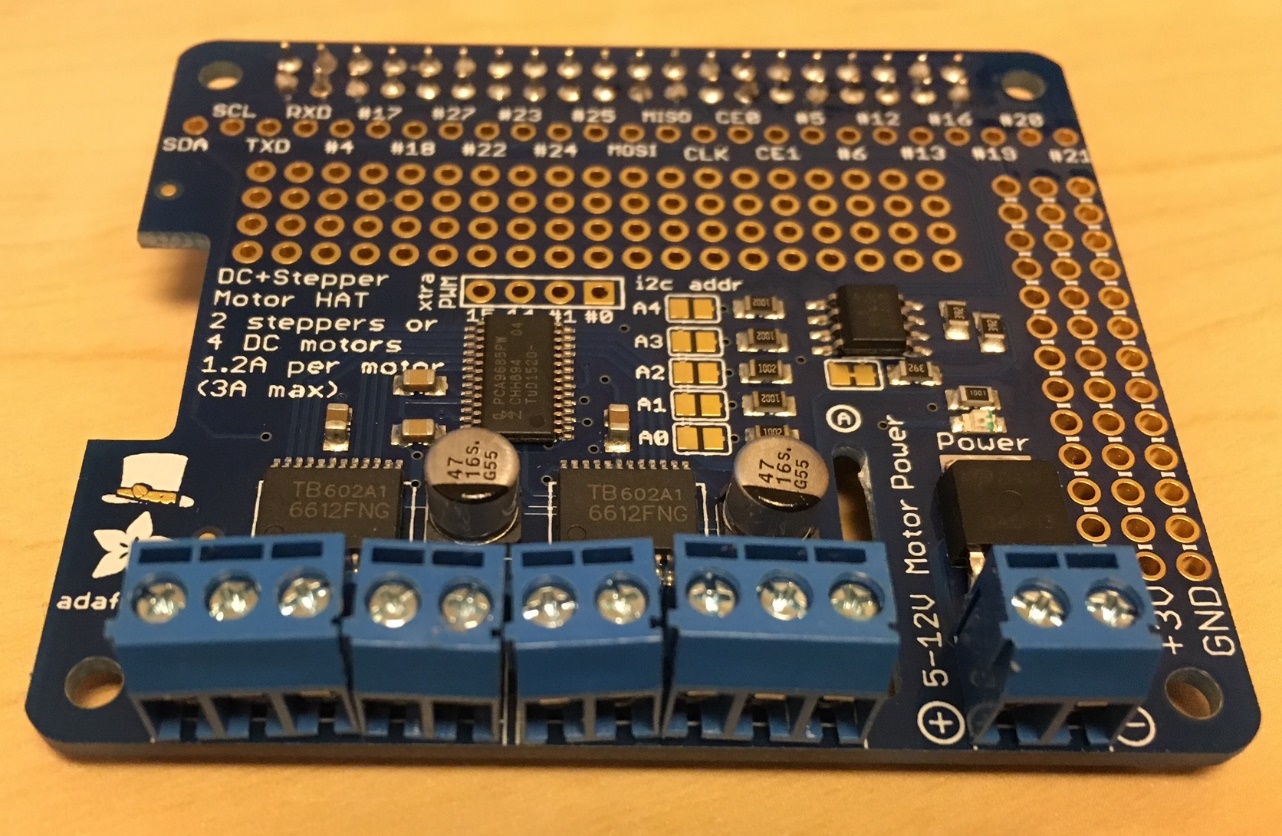
Make sure you write down the IP address of the Pi after connecting it to your network. It will be required later on. If at any point you lose or forget the IP address, the fallback way of figuring out the IP could be to log in to your Wi-Fi router and see the list of all the devices connected to your router. This list should have the IP address information.

# Assembling the hardware

1. Solder the gpio port header on the reverse side of the Motor Hat as shown below



1. Solder the Motor output ports and the Motor Power input ports as shown below

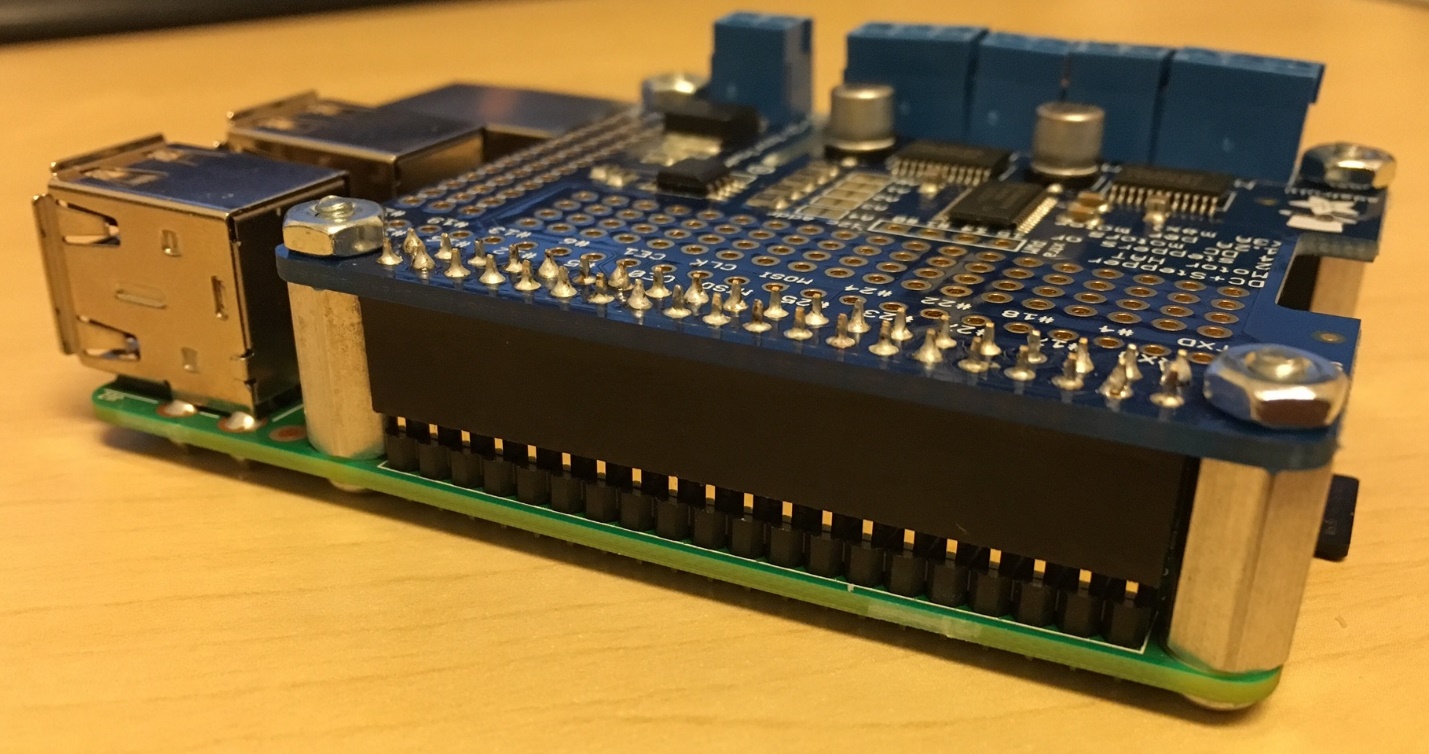


1. Mount the motor hat on top of Raspberry Pi, so that the gpio header is on top of the gpio pins. For added stability, spacers can be added on all four corners of the Pi and the motor hat as shown below. The spacers are held in place with the help of screws and nuts.

Spacers: 8-32 x ½” FxF Aluminum

Screws: #4-40 x ¾” Philips machine screws

Nuts: #4-40 nuts



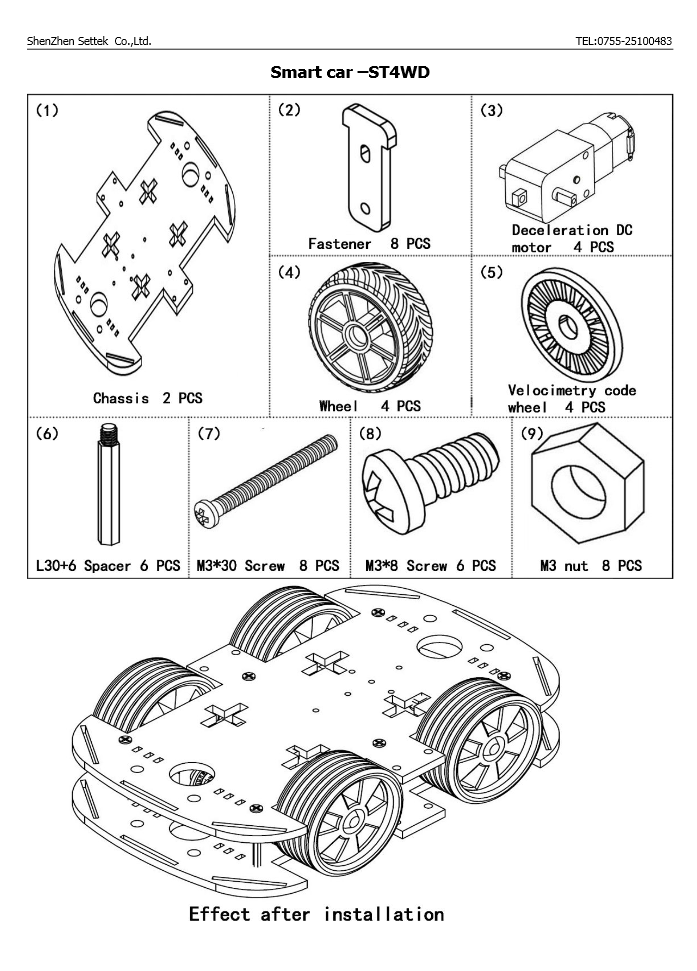
1. Solder the wires to the Motor inputs as shown below

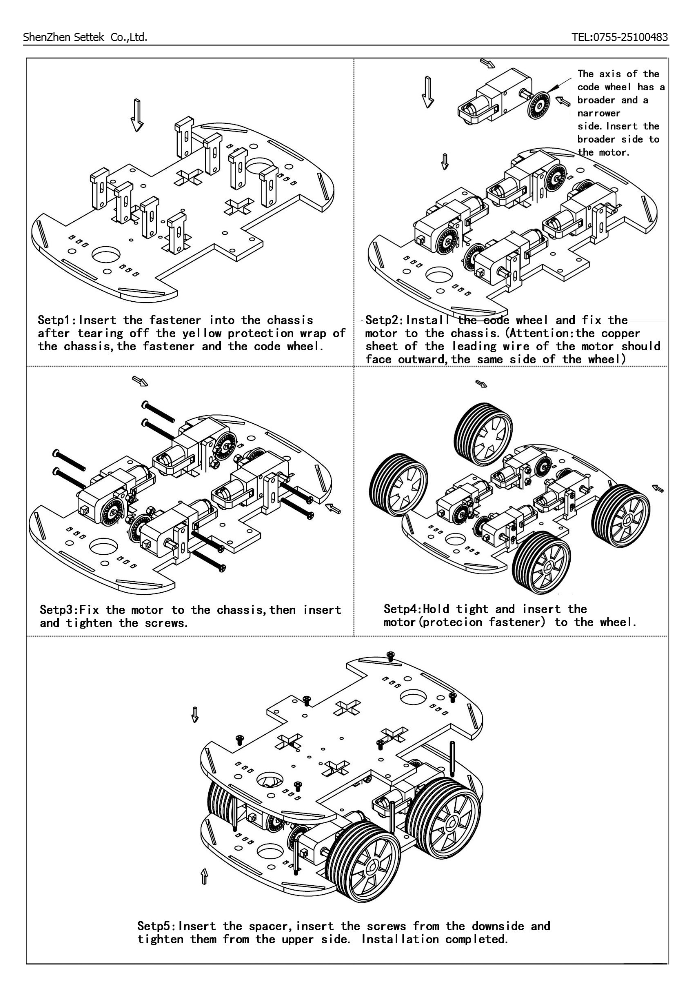


1. Assemble the car chassis as shown [here](http://i.imgur.com/1qPnXQJ.png). The parts info is available [here](http://i.imgur.com/YJ8bu1y.png).

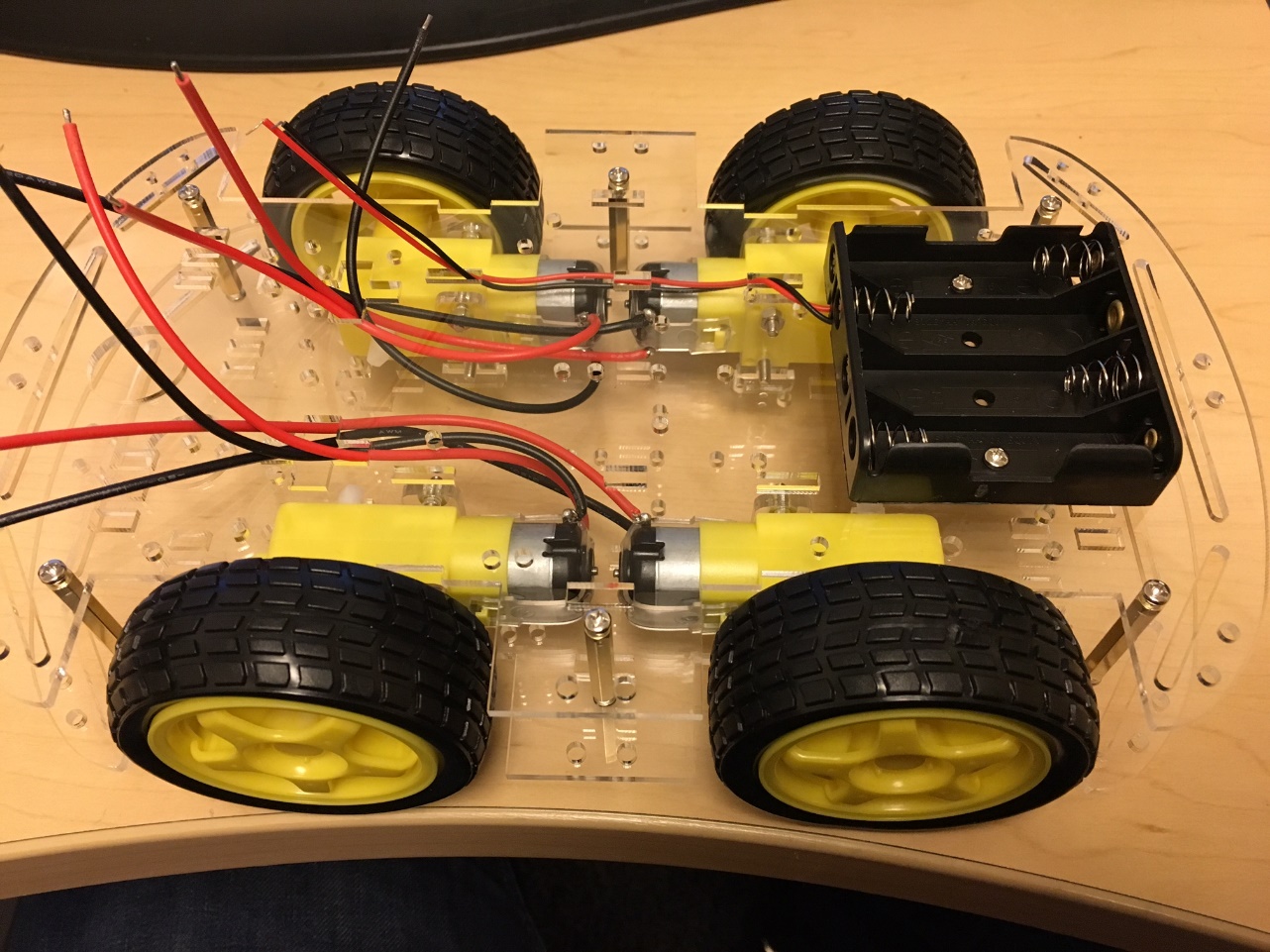
Note: in this project, we do not require the velocity code wheels. Therefore, we can skip the step of adding the code wheels for this project’s purpose.

Follow the instructions shown below to assemble the car chassis. Make sure you remove the brown protection wrap from the acrylic sheets before assembly.





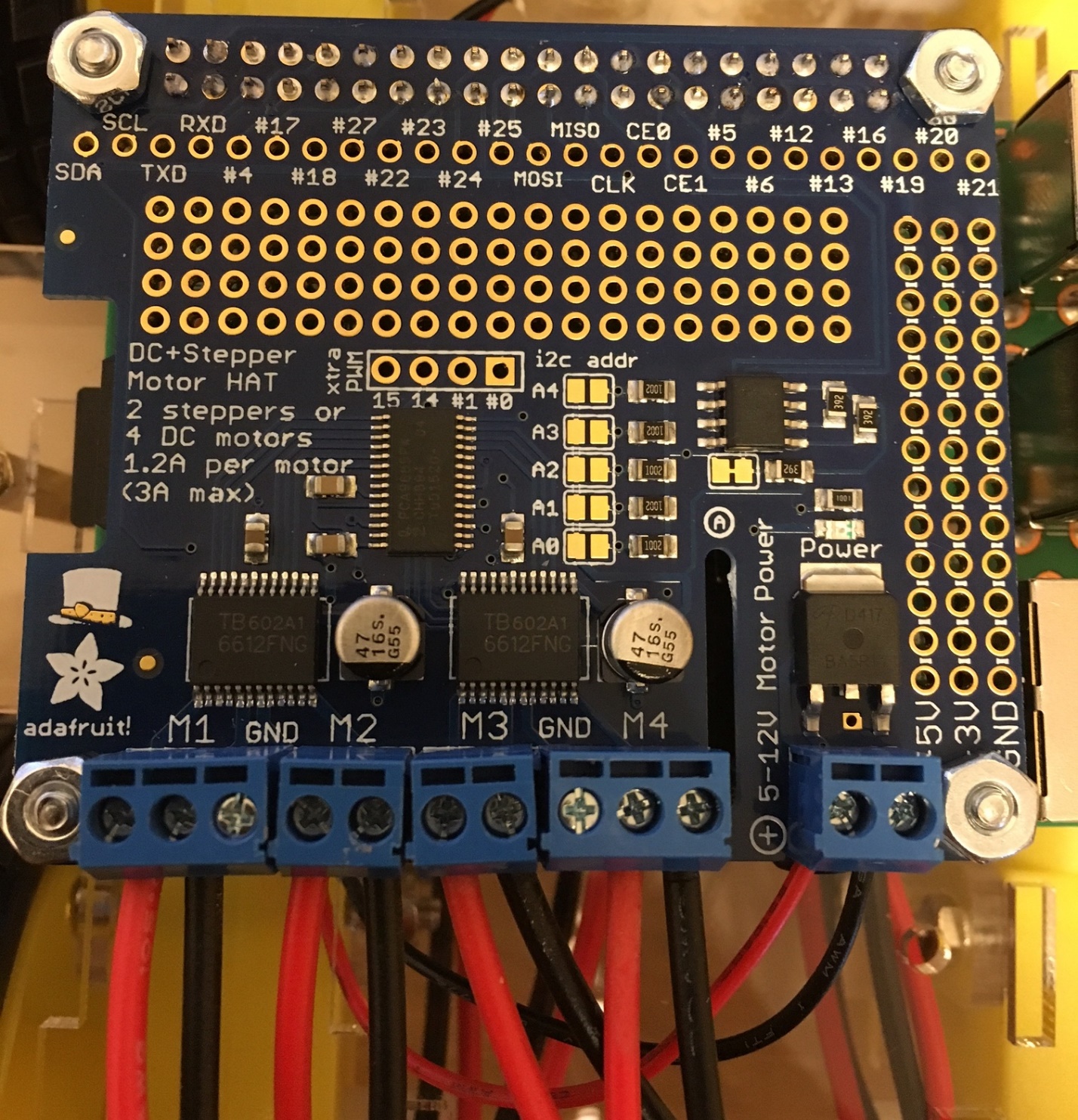
1. Mount the battery holder on top of the car chassis as shown below



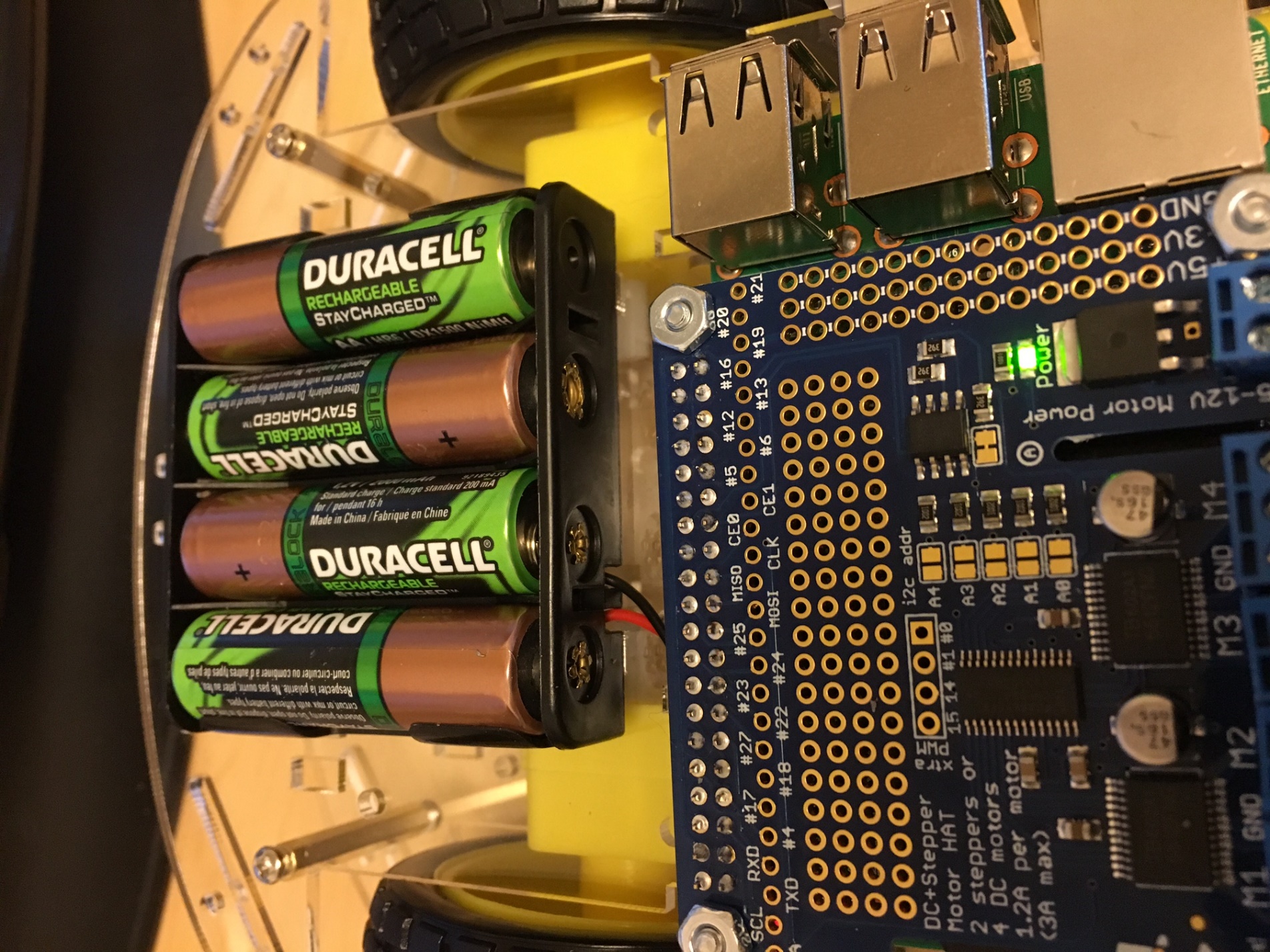
1. Mount the Pi and the motor hat on the car chassis as shown below. (Hint: It can be super glued to the chassis)



1. Connect the wires from the motors to the Motor hat. Connect them so that one red and one black wire from one motor goes to M1. Repeat that for all the other 3 motors. Also connect the output form the battery holder to the Motor Power input on the motor hat. The red wire needs to go in the positive side and the black wire should go in the negative side. The ground (GND) inputs on the motor hat will not connect to anything.



1. Connect the usb power bank to the micro USB power input to the Raspberry Pi. This would power the Raspberry Pi. And put 4 batteries in the battery holder. These batteries will power the motor hat and the motors.



# Designing the Car interface

The first thing we need to do for designing the consumer and the producer apps is to design the car interface, and generate the code using AllJoyn Studio. An AllJoyn interface can have methods (think of them as functions which initiates a process on the producer), properties (read or write properties that can inform or modify the state of a producer) and signals (events that the producer can implement to inform the consumer about a change in state).

We have here a car chassis with 4 wheels. All of these wheels are fixed in place and there is no clear front of rear to the chassis. We need to define a front for our car. To drive forward, we can turn the wheels on the right side in the clockwise direction and the wheels on the left side in the counter-clockwise direction, all at the same speed. Vice-versa to drive in the reverse. To turn left or right, we need to implement – differential drive. Therefore, to turn left, all the wheels would turn clockwise at the same speed. And to turn right, all the wheels will turn counter-clockwise at the same speed.

So to drive the car we need to send three things from the consumer to the producer.

1. When to start driving, and when to stop.
2. In which direction to drive the car.
3. At what speed to drive the car.

A sample interface is shown below that implements the above design. Here we have a method called Drive that takes in two arguments – a bool type to signal whether to start or stop the car, and an uint type which tells the direction to drive. The uint value can be mapped to the directions in an enumeration on both the producer and the consumer side. We also have a read/write property called Speed, which can set and get the speed of the car from the producer. We have a third property called State, which is a read-only property that can be used by the consumer to get the on/off and the direction state of the car at any time. This could be used to update the UI of the consumer app.

<node name="/com/microsoft/Samples">

<interface name="com.microsoft.Sample.AllJoynCar">

<method name="Drive">

<arg name="OnOff" type="b" direction="in" />

<arg name="Direction" type="u" direction="in" />

</method>

<property name="Speed" type="u" access="readwrite" />

<property name="State" type="(bu)" access="read">

<annotation name="org.freedesktop.DBus.Property.EmitsChangedSignal" value="true" />

</property>

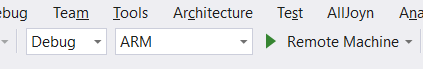
</interface>

</node>

The sample producer and the consumer apps implementing the above interface can be found [here](https://github.com/rajat-rastogi/InternBootcamp/tree/master/AllJoynCar).

# Deploying and running the apps

1. Power on the Car by connecting the power to Raspberry Pi and inserting the batteries in the battery holder.
2. Make sure your machine/laptop is on the same network as the one we configured in the “Connect to the network section above”.
3. Get the apps from the [intern bootcamp github](https://github.com/rajat-rastogi/InternBootcamp/tree/master/AllJoynCar) repository.
4. Open the Producer app in Visual Studio.
5. Select ARM as your solution platform and Remote Machine as your target device.
6. Enter the IP address of the Pi, and make sure that the Authentication mode is set to Universal (Unencrypted Protocol)



These settings can also be re-visited by selecting the project in the solution explorer and selecting Project Menu🡪Properties (last option in the drop down)🡪Debug

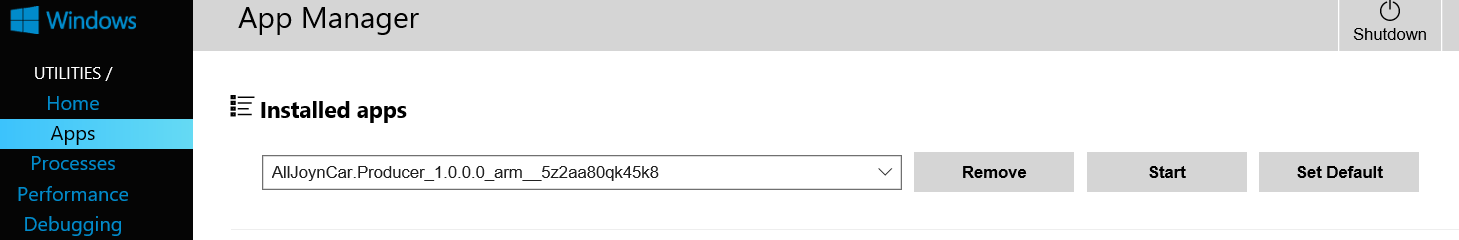
1. Deploy.
2. Once visual studio says that the app is deployed successfully. Stop debugging.
3. Open browser and navigate to the following address – http://<IP Address>:8080. So for example the IP address of the Pi is 192.168.2.1, we would navigate to <http://192.168.2.1:8080>
4. This will prompt for credentials. The default credentials are:

Username: Administrator

Password: p@ssw0rd

See the [Windows Device Portal](https://developer.microsoft.com/en-us/windows/iot/win10/tools/deviceportal) page for more details.

1. Click Apps on the left hand side panel.
2. Under the Installed apps drop down select the AllJoynCar.Producer app and click the “Set Default” button. This will set the AllJoyn producer app as the default startup app on boot.



1. Reboot the Pi by clicking Restart on the top right corner of the device portal. Once the Pi boots again it should launch the AllJoyn.Producer app. This can be confirmed by refreshing the device portal after few minutes to make sure that the AllJoyn.Producer app is running.
2. Now open the consumer app in Visual Studio. Either deploy it on the local machine or deploy it phone. Once the app is running, it should connect to the Car in few seconds. Give the app a minute to find the car and connect to it.
3. Start driving the car. ☺

If you make any changes to the Producer app and want to re-deploy the newer version of the app, then make sure you set the IoTCoreDefaultApp as the default app on the Pi first and them attempt to re-deploy from Visual studio. If AllJoyn.Producer is set as the current default app, Visual studio will throw errors trying to deploy again. Once re-deployed, you can repeat the above steps to make the producer app as the default app again.

# Links and references

1. A lot of reference on Windows IoT core can be found on <http://www.windowsondevices.com/>
2. Check out the [Docs and sample page](https://developer.microsoft.com/en-us/windows/iot/win10/StartCoding.htm) for more documentation and some other cool samples you can try out with Windows IoT core on Raspberry Pi.
3. Github repository - <https://github.com/rajat-rastogi/InternBootcamp>
4. Maker Garage – Building 27/1311 – has few soldering irons and other tools you might require to assemble the hardware. They have the spacers, nuts and screws as well. Feel free to drop by during your internship and assemble your hardware.
5. Feel free to contact me with any questions you may have – [rajatras@microsoft.com](mailto:rajatras@microsoft.com)