



Lab 1: Managing Sensors and Actuators with Raspberry Pi & Arduino using Wyliodrin Studio (1)

物聯網技術與應用(英) IoT/M2M Technologies and Applications

國立交通大學資訊工程系
Department of Computer Science
National Chiao Tung University

September 21, 2018

Outline

- Wyliodrin Studio Overview
 - Setup and Starting Wyliodrin Studio
 - Wyliodrin Studio GUI *(Checkpoint 1)*
- Raspberry and Arduino Basics
- Hello World Application *(Checkpoint 2)*
- Traffic Light Simulator Application
 - Assembling Schematics *(Checkpoint 3)*
 - Coding with Blocks *(Checkpoint 4)*



WYLIODRIN STUDIO OVERVIEW

Wyliodrin Studio Overview (1)

- Wyliodrin Studio is an Open Source, local version of Wyliodrin (<https://www.wyliodrin.com/>).
- Wyliodrin STUDIO is an standalone IDE for software and hardware development for IoT and Embedded Linux systems.
- You can program and monitor the boards which are in the same network or are directly connected to your computer.



Wyliodrin Studio Overview (2)

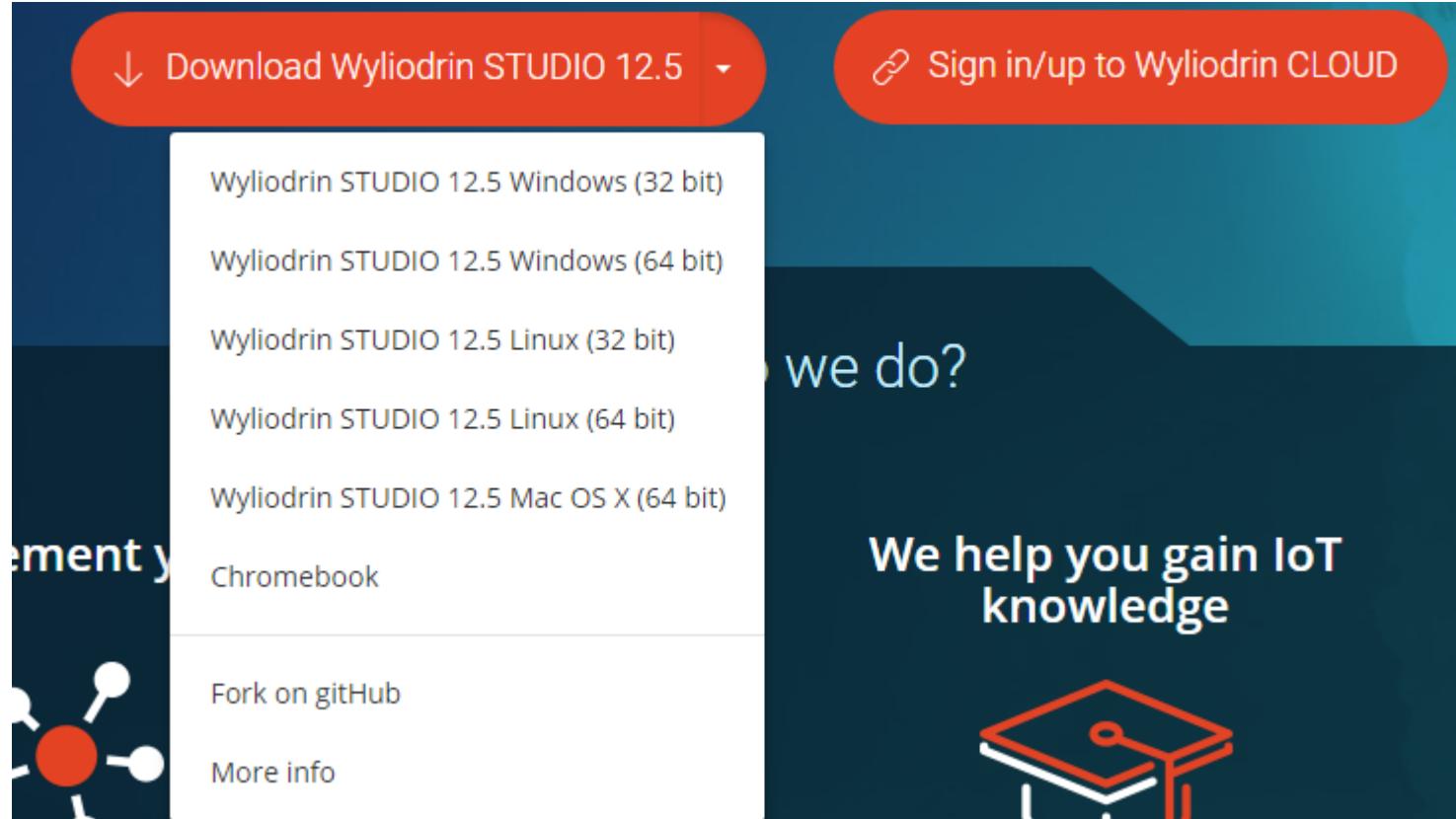
- Connect to devices using TCP/IP or serial port
- Develop software and firmware for IoT in several programming languages
- Shell access to the device
- Import and export Wyliodrin STUDIO projects
- Visual dashboard for displaying sensor data
- Display the hardware schematics
- Manage packages for Python and Javascript
- Task manager for managing the device
- Network connection manager for the device (Ethernet and WiFi)
- Interactive electronics documentation (resistor color code)
- Example projects and firmware
- Wyliodrin API documentation in C/C++, Python and Javascript

Wyliodrin Studio Overview (3)

- **Supported devices**
 - UDOO Neo
 - **Raspberry Pi and Arduino**
 - BeagleBone Black
 - **Arduino**
- **Supported languages**
 - **Visual Programming (translates to Python)**
 - Javascript
 - Python
 - Shell Script (bash)

Wyliodrin Studio Setup

<https://www.wyliodrin.com/>



The screenshot shows the Wyliodrin Studio download page. On the left, there's a sidebar with a red header containing a download link and a sign-in/up link. Below the header, there's a list of download options for different operating systems and devices. On the right, there's a large blue background area with text and an icon.

- Download Wyliodrin STUDIO 12.5
- Sign in/up to Wyliodrin CLOUD

- Wyliodrin STUDIO 12.5 Windows (32 bit)
- Wyliodrin STUDIO 12.5 Windows (64 bit)
- Wyliodrin STUDIO 12.5 Linux (32 bit)
- Wyliodrin STUDIO 12.5 Linux (64 bit)
- Wyliodrin STUDIO 12.5 Mac OS X (64 bit)
- Chromebook
- Fork on GitHub
- More info

we do?

We help you gain IoT knowledge



Attention!! Wyliodrin Studio is already installed in the “IoTClass” VM.
You don't need to execute these steps.

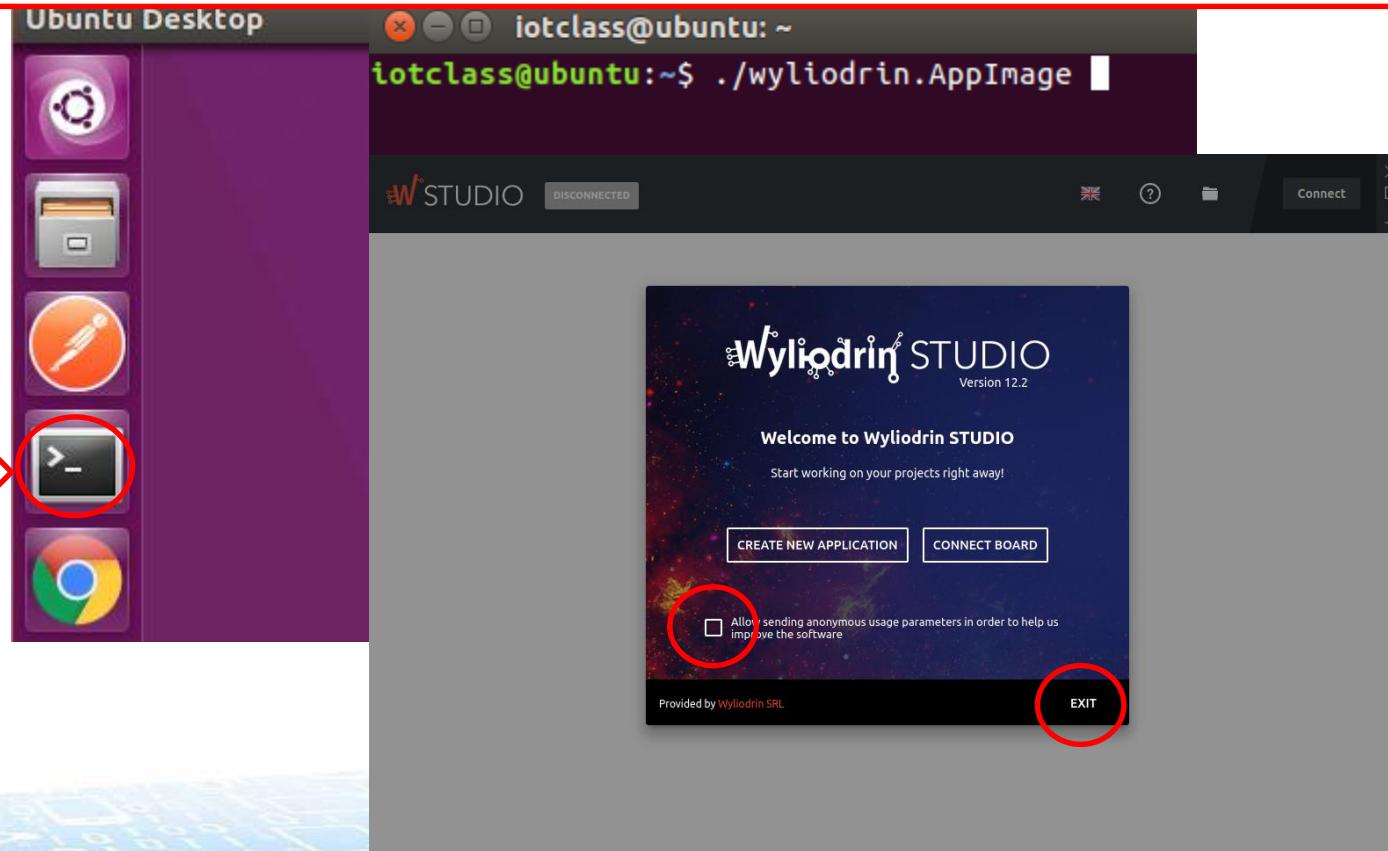
Use your VM from now on...

The steps showed in the coming slides have to be performed in your VM.

Username = iotclass
Password = iotclass

Starting Wyliodrin Studio

1. Open a Terminal window.
2. Type ./wyliodrin.AppImage and press ENTER.
3. **Attention!! Uncheck “Allow sending anonymous” and Click Exit (If needed). Otherwise you will see just an empty window, which is correct!**



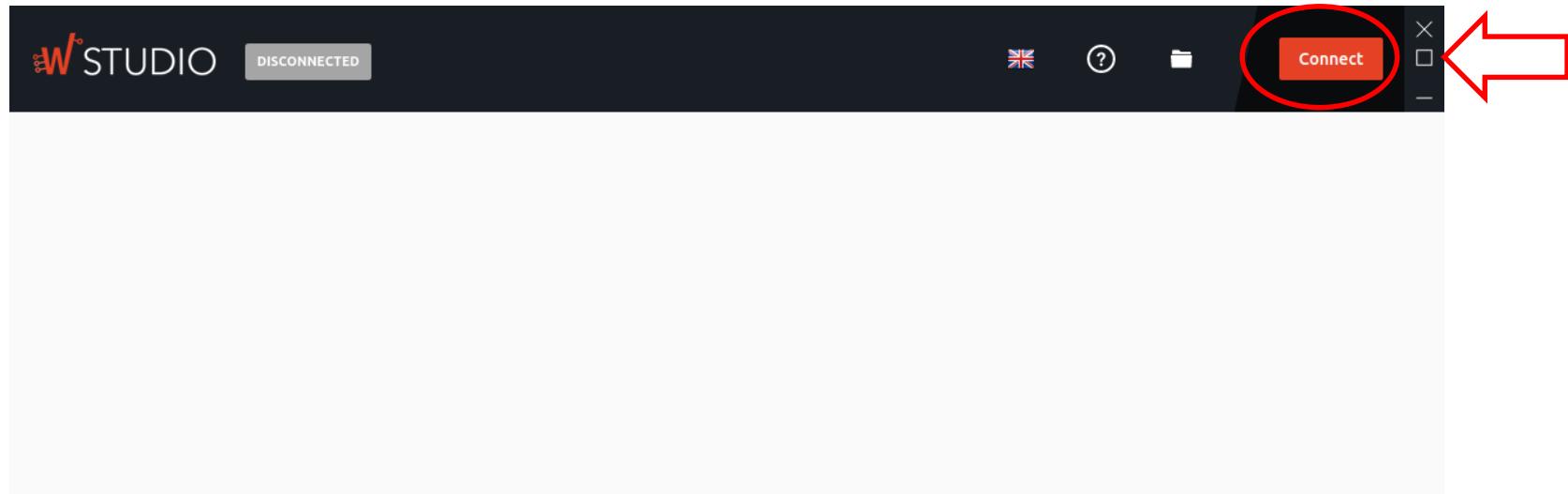
Connect the Raspberry Pi to the Power Source

Connect your Raspberry Pi to the Power Source Now!

Connecting a Raspberry Pi to Wyliodrin Studio (1)

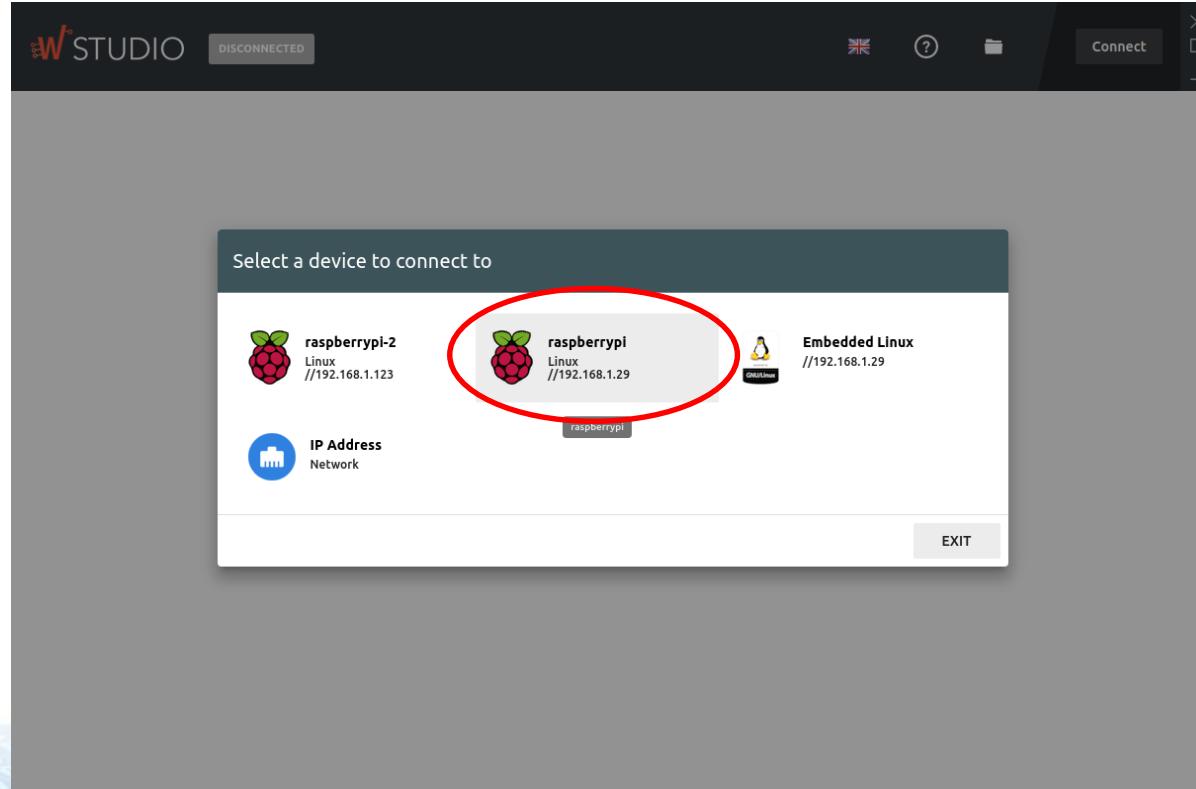
In order to see all the options in the GUI, we need to connect Wyliodrin with a board. The following steps are used to connect Wyliodrin with a Raspberry Pi.

1. Click on “Connect”.



Connecting a Raspberry Pi to Wyliodrin Studio (2)

2. Click on the Raspberry Pi **ASSIGNED TO YOUR TEAM**.
Use the IP address instructed by TA as guide.
(Ask TA if you are not sure which Pi belongs to your team).



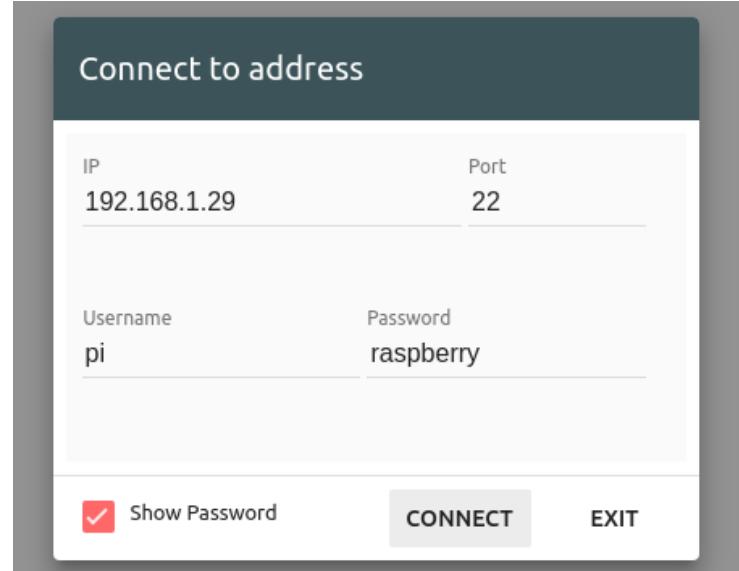
Connecting a Raspberry Pi to Wyliodrin Studio (3)

6. Input the following credentials:

Username = pi

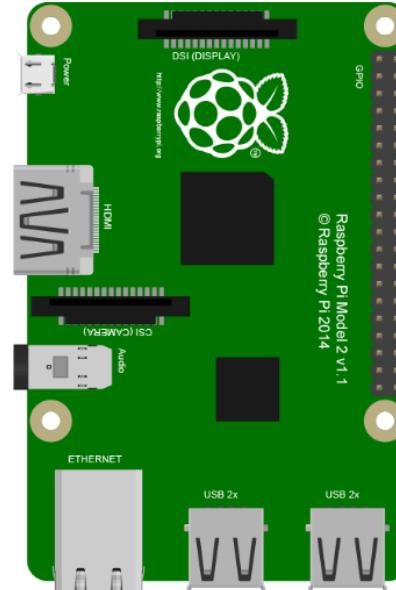
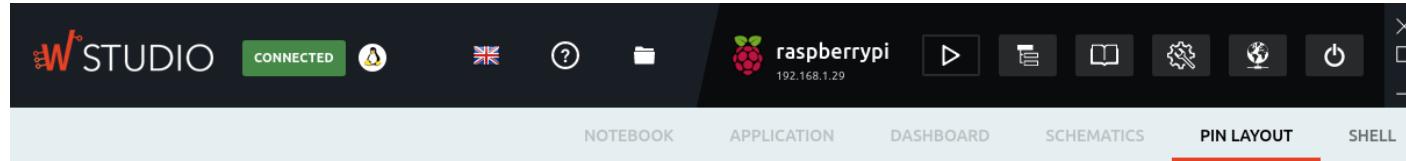
Password = raspberry

7. Click on “Connect”.



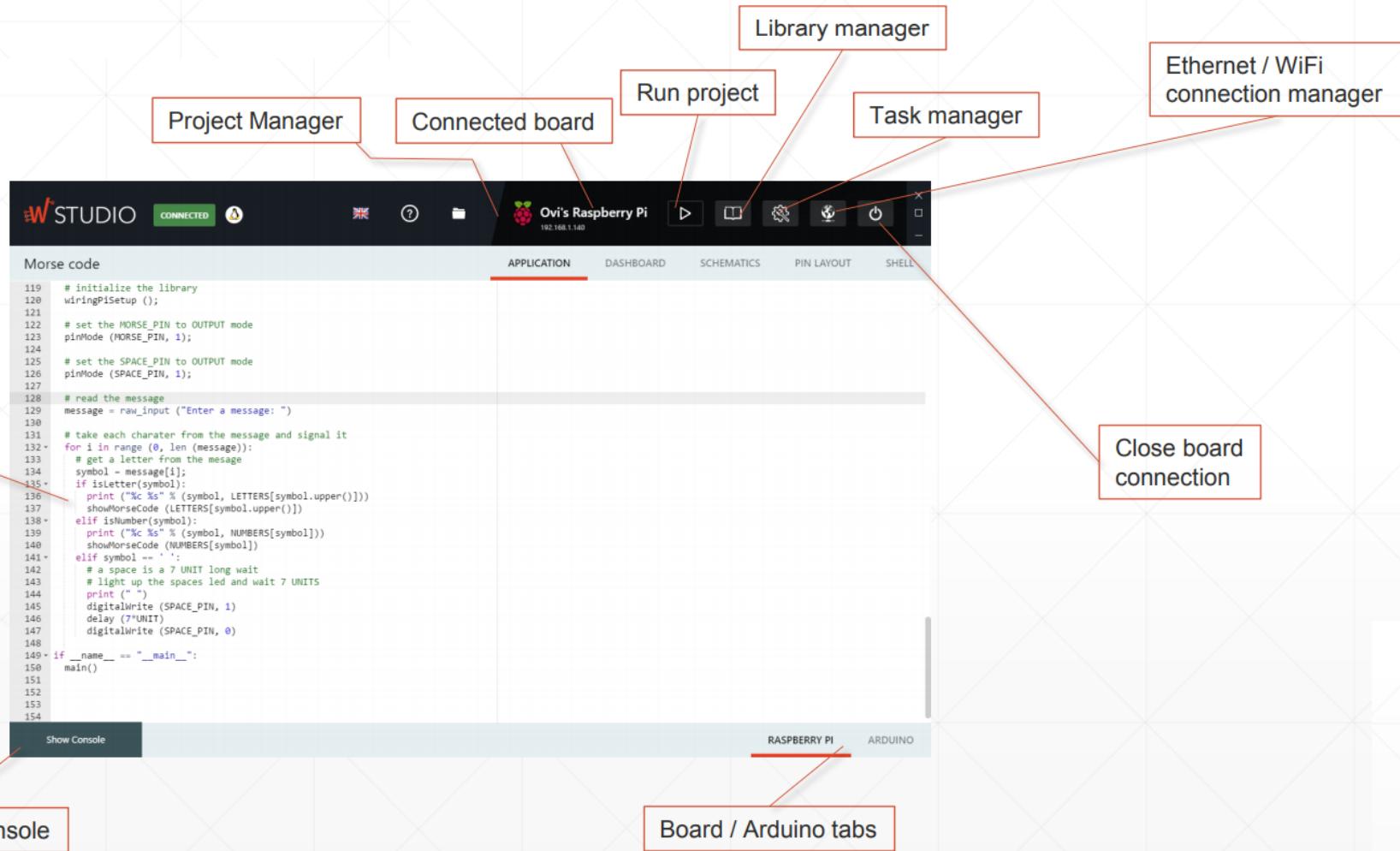
Connecting a Raspberry Pi to Wyliodrin Studio (4)

You should see now all the features that Wyliodrin Studio GUI offers.



Raspberry Pi 2 Model B (J8 Header) pinout					
WiringPi	BCM(Name)	Physical	Physical	BCM(Name)	WiringPi
	3v3 Power	1	1	5v Power	
8	BCM 2 (SDA)	2	2	5v Power	
9	BCM 3 (SCL)	3	3	Ground	
7	BCM 4 (GPCLK0)	4	4	BCM 14 (TXD)	15
	Ground	5	5	BCM 15 (RXD)	16
0	BCM 17	6	6	BCM 18 (PCM_C)	1
2	BCM 27 (PCM_D)	7	7		
3	BCM 22	8	8	Ground	
	3v3 Power	9	9	BCM 23	4
12	BCM 10 (MOSI)	10	10	BCM 24	5
13	BCM 9 (MISO)	11	11		
14	BCM 11 (SCLK)	12	12	Ground	
	Ground	13	13	BCM 25	6
	BCM 0 (ID_SD)	14	14	BCM 8 (CE0)	10
21	BCM 5	15	15	BCM 7 (CE1)	11
22	BCM 6	16	16	BCM 1 (ID_SC)	
23	BCM 13	17	17	Ground	
24	BCM 19 (MISO)	18	18	BCM 12	26
25	BCM 26	19	19	Ground	
	Ground	20	20	BCM 16	27
		21	21	BCM 20 (MOSI)	28
		22	22	BCM 21 (SCLK)	29

Wyliodrin Studio GUI (1)



Wyliodrin Studio GUI (2)

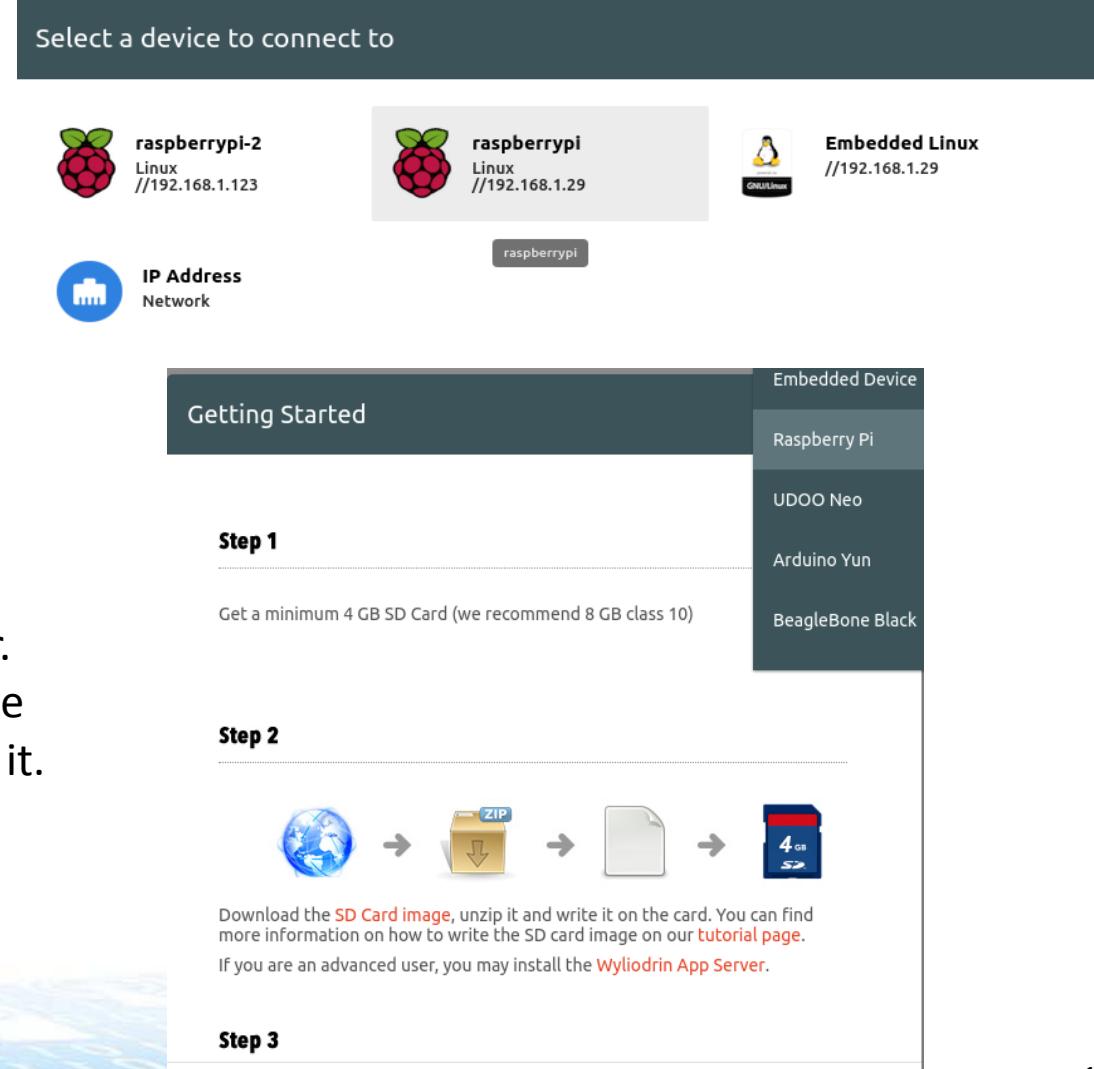
Board Connection

- Direct connection (Serial communication).
- Remote connection (Uses mDNS to discover devices in the same network).

Additional Software

- Any board needs to run an instance of wyliodrin-server.
- Download the SD card image for your board before using it.
(TA has done this for you!)

Select a device to connect to



The interface shows a list of connected devices:

- raspberrypi-2 (Linux //192.168.1.123)
- raspberrypi (Linux //192.168.1.29)
- Embedded Linux (GNU/Linux //192.168.1.29)

Below the devices, there's a sidebar titled "Getting Started" with a "Step 1" section:

Step 1
Get a minimum 4 GB SD Card (we recommend 8 GB class 10)

To the right of the sidebar, a vertical menu lists supported boards:

- Embedded Device
- Raspberry Pi
- UDOO Neo
- Arduino Yun
- BeagleBone Black

Under "Step 2", there's a diagram showing the process from a globe icon to a ZIP file icon, then to a document icon, and finally to an SD card icon.

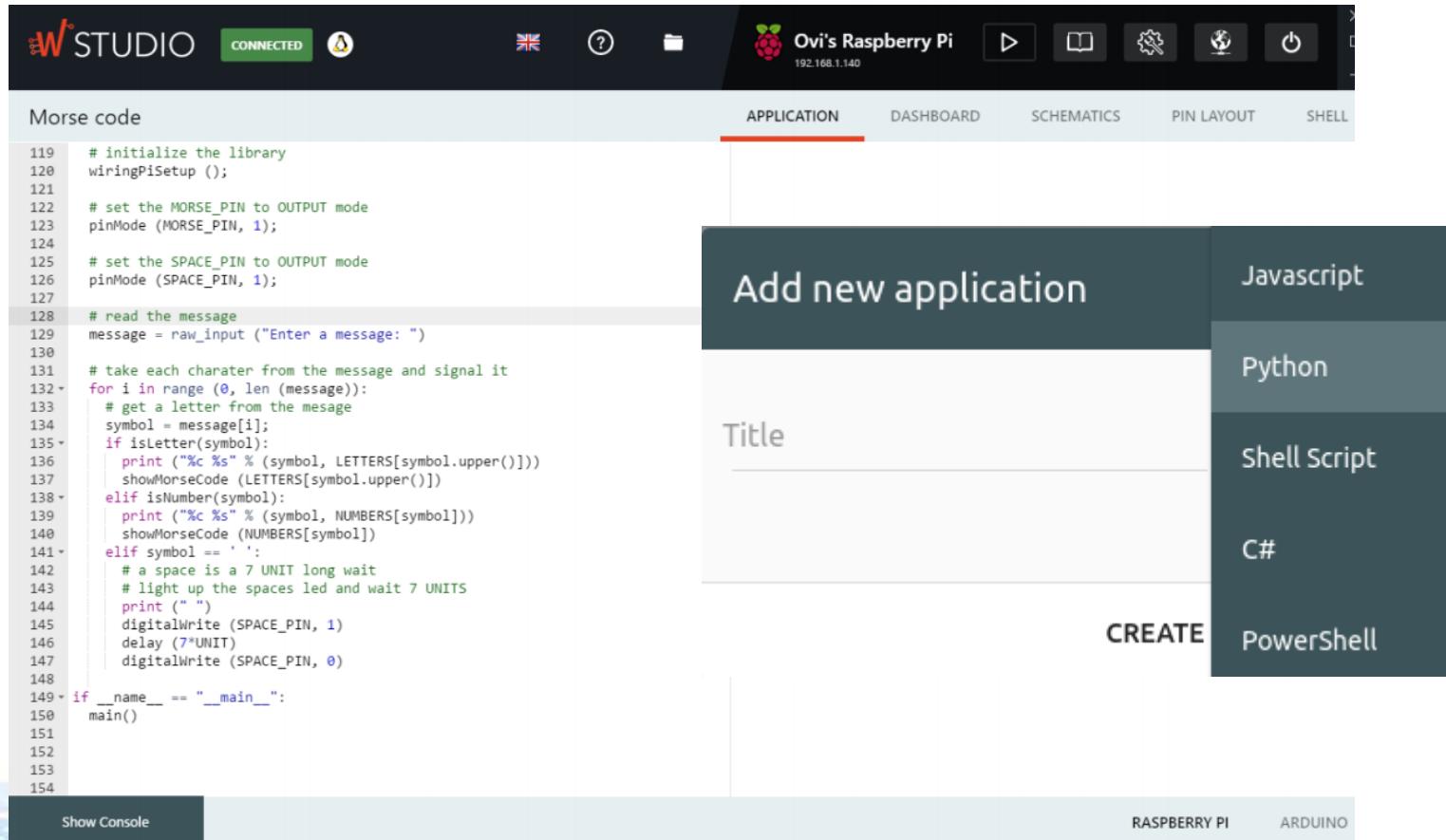
Text below the diagram reads:
Download the [SD Card image](#), unzip it and write it on the card. You can find more information on how to write the SD card image on our [tutorial page](#).
If you are an advanced user, you may install the [Wyliodrin App Server](#).

Step 3

Wyliodrin Studio GUI (3)

Programming

- Professional code editor.
- Advanced features such as autocomplete.



The screenshot shows the Wyliodrin Studio interface. At the top, there's a header bar with the Wyliodrin logo, connection status (CONNECTED), a gear icon, and the text "Ovi's Raspberry Pi" with IP address "192.168.1.140". Below the header are tabs: APPLICATION (which is selected), DASHBOARD, SCHEMATICS, PIN LAYOUT, and SHELL. The main area has a title "Morse code" and displays the following Morse code conversion code:

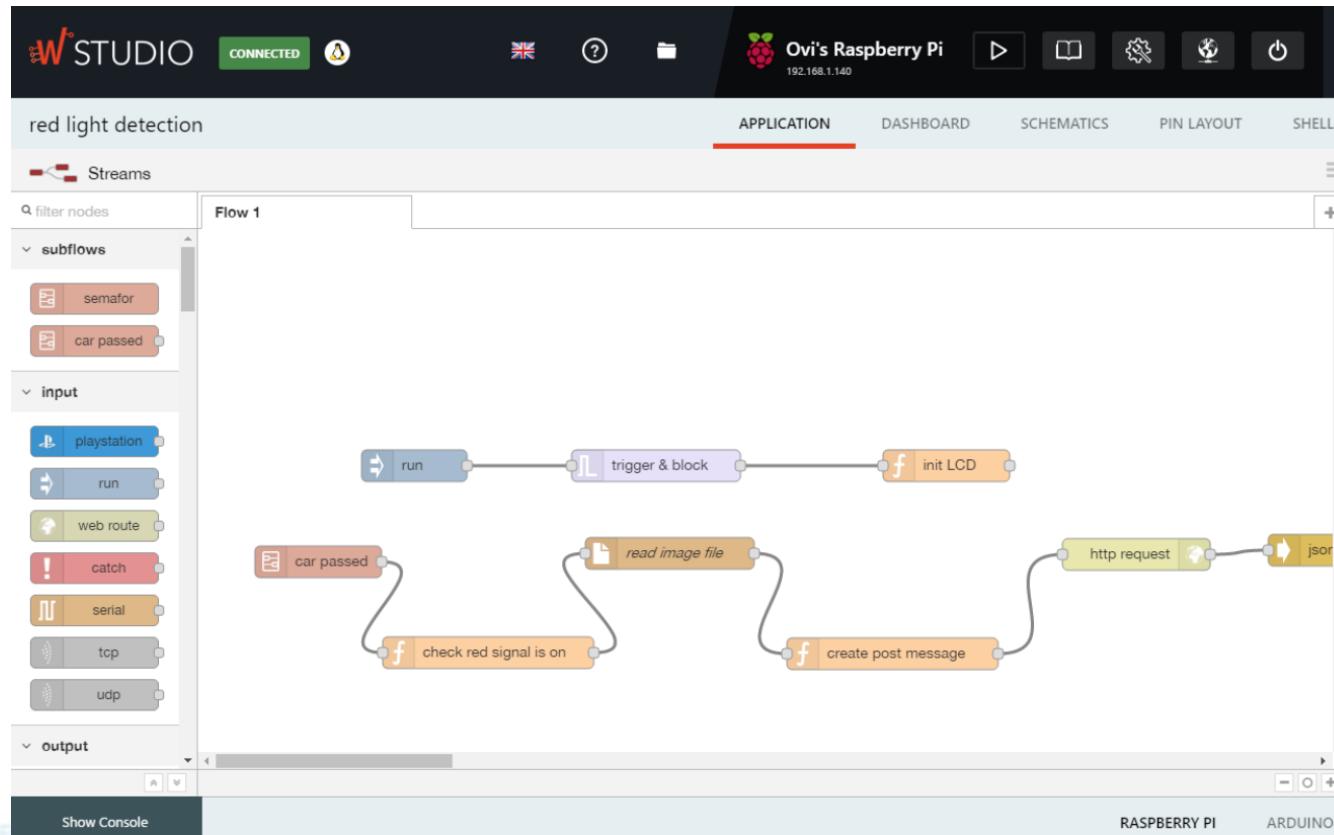
```
119 # initialize the library
120 wiringPiSetup ();
121
122 # set the MORSE_PIN to OUTPUT mode
123 pinMode (MORSE_PIN, 1);
124
125 # set the SPACE_PIN to OUTPUT mode
126 pinMode (SPACE_PIN, 1);
127
128 # read the message
129 message = raw_input ("Enter a message: ")
130
131 # take each character from the message and signal it
132 for i in range (0, len (message)):
133     # get a letter from the message
134     symbol = message[i];
135     if isLetter(symbol):
136         print ("%c %s" % (symbol, LETTERS[symbol.upper()]))
137         showMorseCode (LETTERS[symbol.upper()])
138     elif isNumber(symbol):
139         print ("%c %s" % (symbol, NUMBERS[symbol]))
140         showMorseCode (NUMBERS[symbol])
141     elif symbol == ' ':
142         # a space is a 7 UNIT long wait
143         # light up the spaces led and wait 7 UNITS
144         print (" ")
145         digitalWrite (SPACE_PIN, 1)
146         delay (7*UNIT)
147         digitalWrite (SPACE_PIN, 0)
148
149 if __name__ == "__main__":
150     main()
151
152
153
154
```

To the right of the code editor, a modal dialog titled "Add new application" is open. It has a "Title" input field and a "CREATE" button. To the right of the modal, there's a sidebar with options: Javascript, Python, Shell Script, C#, and PowerShell.

Wyliodrin Studio GUI (4)

Streams

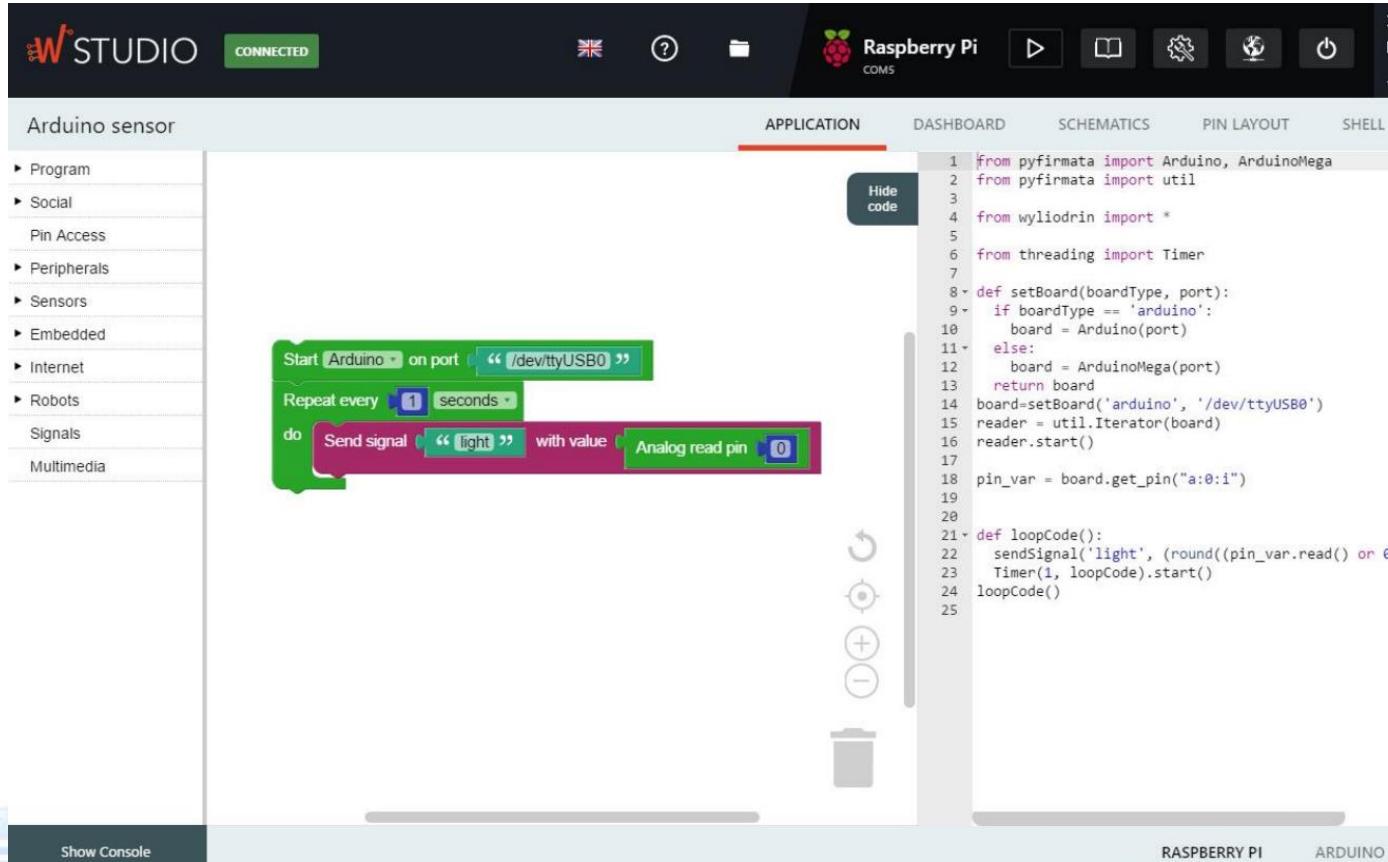
- Data flow programming.
- Implementation of node-red.



Wyliodrin Studio GUI (5)

Visual Programming

- Drag and drop blocks of code.
- Implements Google Blockly.
- View as Python code gets generated.



The screenshot shows the Wyliodrin Studio interface. At the top, there's a toolbar with icons for file operations, Raspberry Pi connection status (CONNECTED), and various settings. Below the toolbar, the main window has tabs for APPLICATION, DASHBOARD, SCHEMATICS, PIN LAYOUT, and SHELL. The APPLICATION tab is selected. On the left, a sidebar menu under 'Arduino sensor' lists categories like Program, Social, Pin Access, Peripherals, Sensors, Embedded, Internet, Robots, Signals, and Multimedia. In the center, a workspace displays a sequence of blocks: 'Start Arduino on port "/dev/ttyUSB0"', 'Repeat every 1 seconds', 'do Send signal "light" with value Analog read pin 0'. To the right of the workspace, the generated Python code is shown:

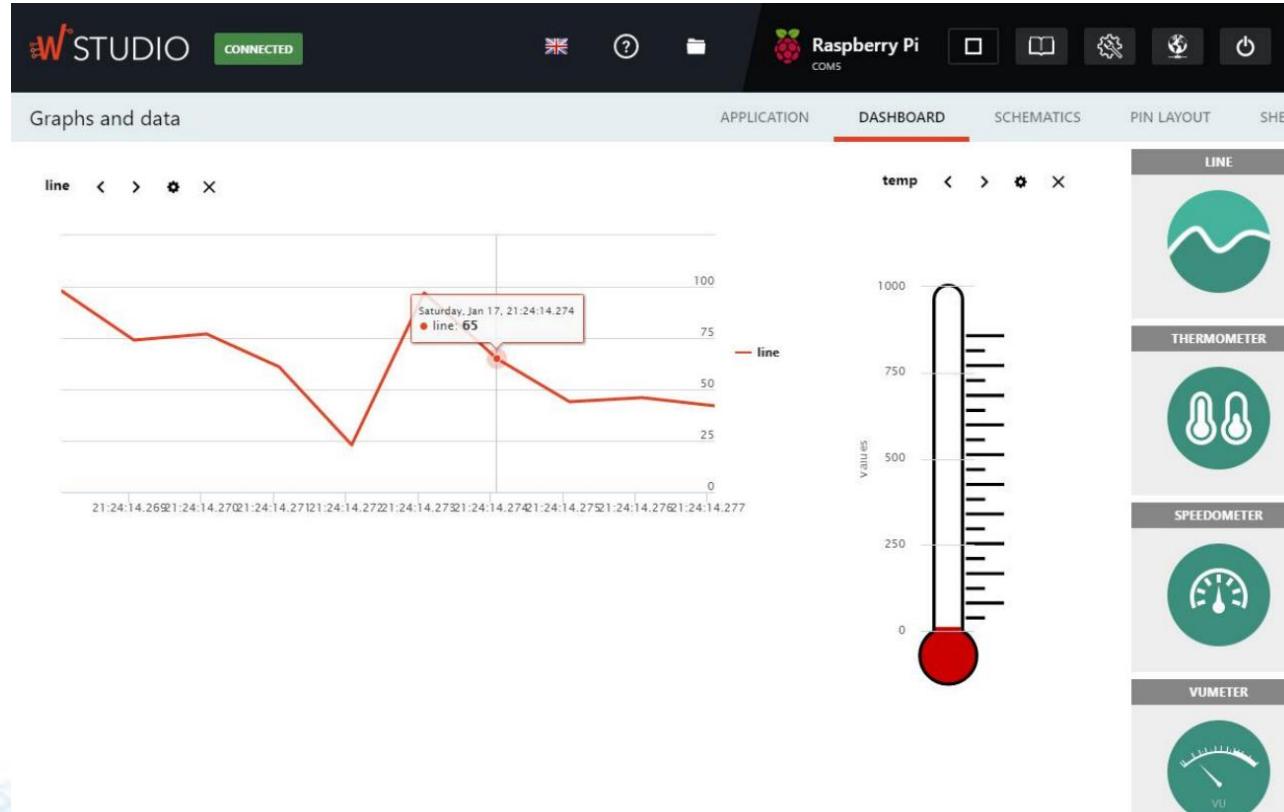
```
1 from pyfirmata import Arduino, ArduinoMega
2 from pyfirmata import util
3
4 from wyliodrin import *
5
6 from threading import Timer
7
8 def setBoard(boardType, port):
9     if boardType == 'arduino':
10         board = Arduino(port)
11     else:
12         board = ArduinoMega(port)
13     return board
14
15 board=setBoard('arduino', '/dev/ttyUSB0')
16 reader = util.Iterator(board)
17 reader.start()
18
19 pin_var = board.get_pin("a:0:i")
20
21 def loopCode():
22     sendSignal('light', (round((pin_var.read() or 0)
23     Timer(1, loopCode).start()
24 loopCode()
```

At the bottom, there are buttons for Show Console, RASPBERRY PI, and ARDUINO.

Wyliodrin Studio GUI (6)

Debug

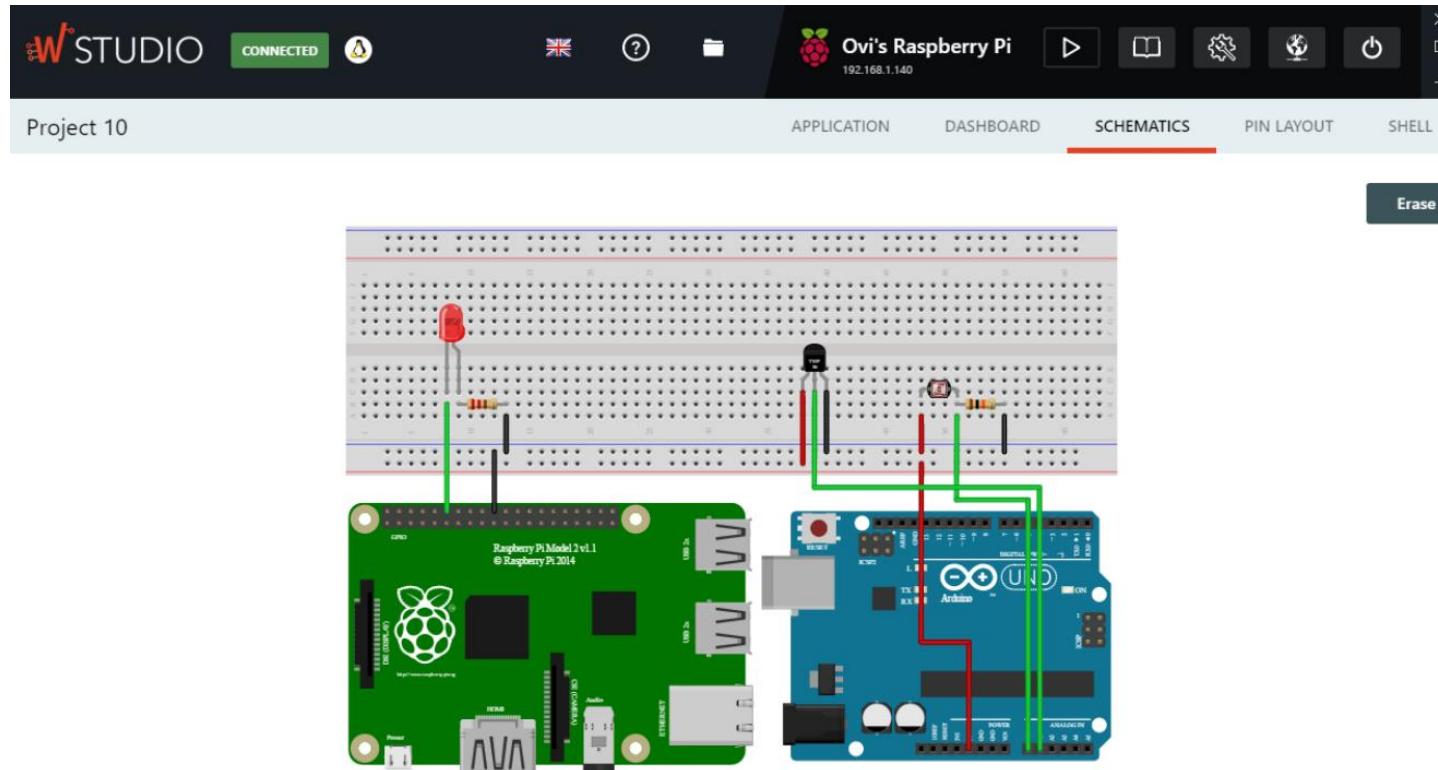
- Send signals to dashboard.
- Real time graphs allowing marks.



Wyliodrin Studio GUI (7)

Fritzing Schemas

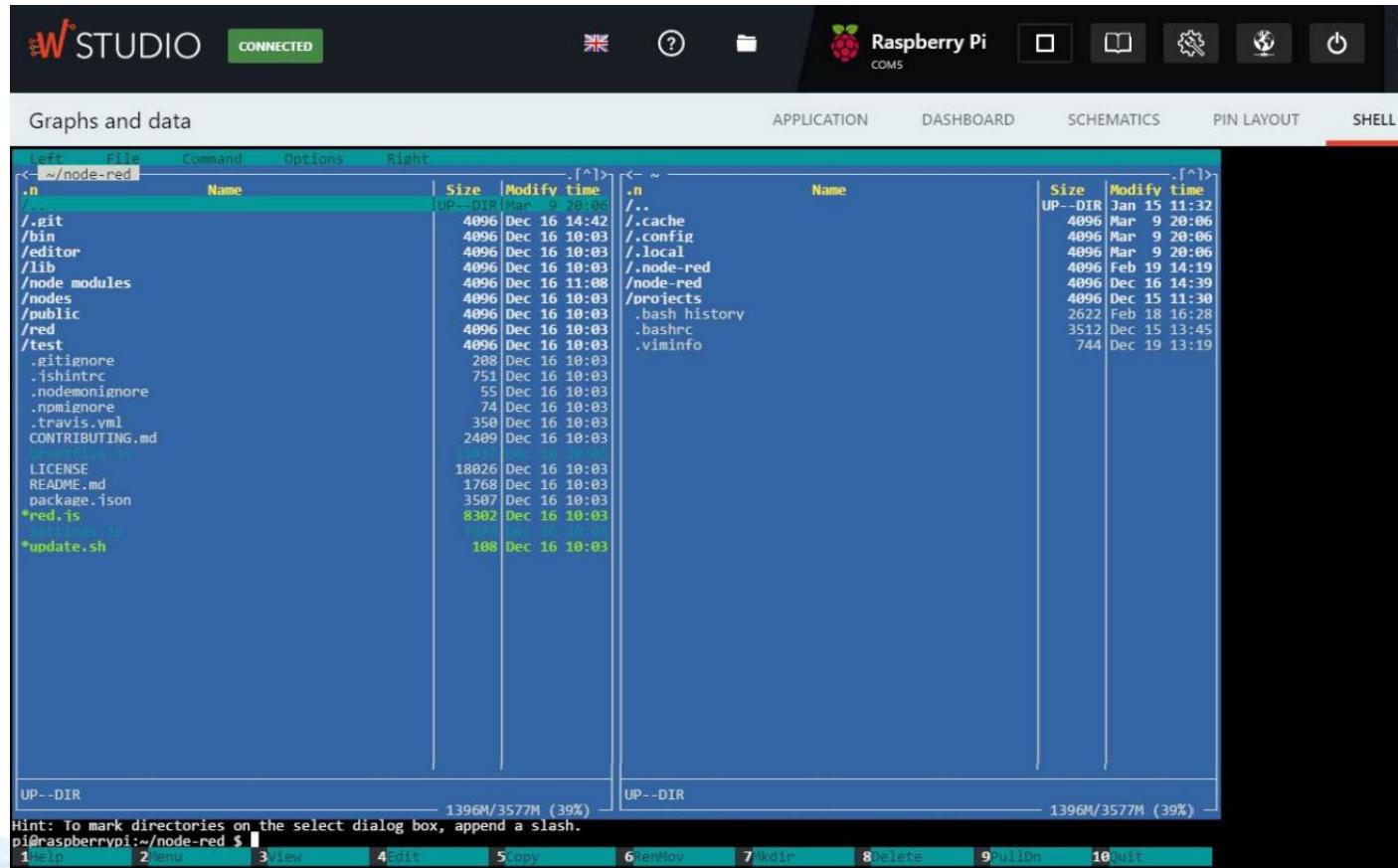
- Import SVG from Fritzing (<http://fritzing.org>).
- Attach schema to application (helps on documentation).



Wyliodrin Studio GUI (7)

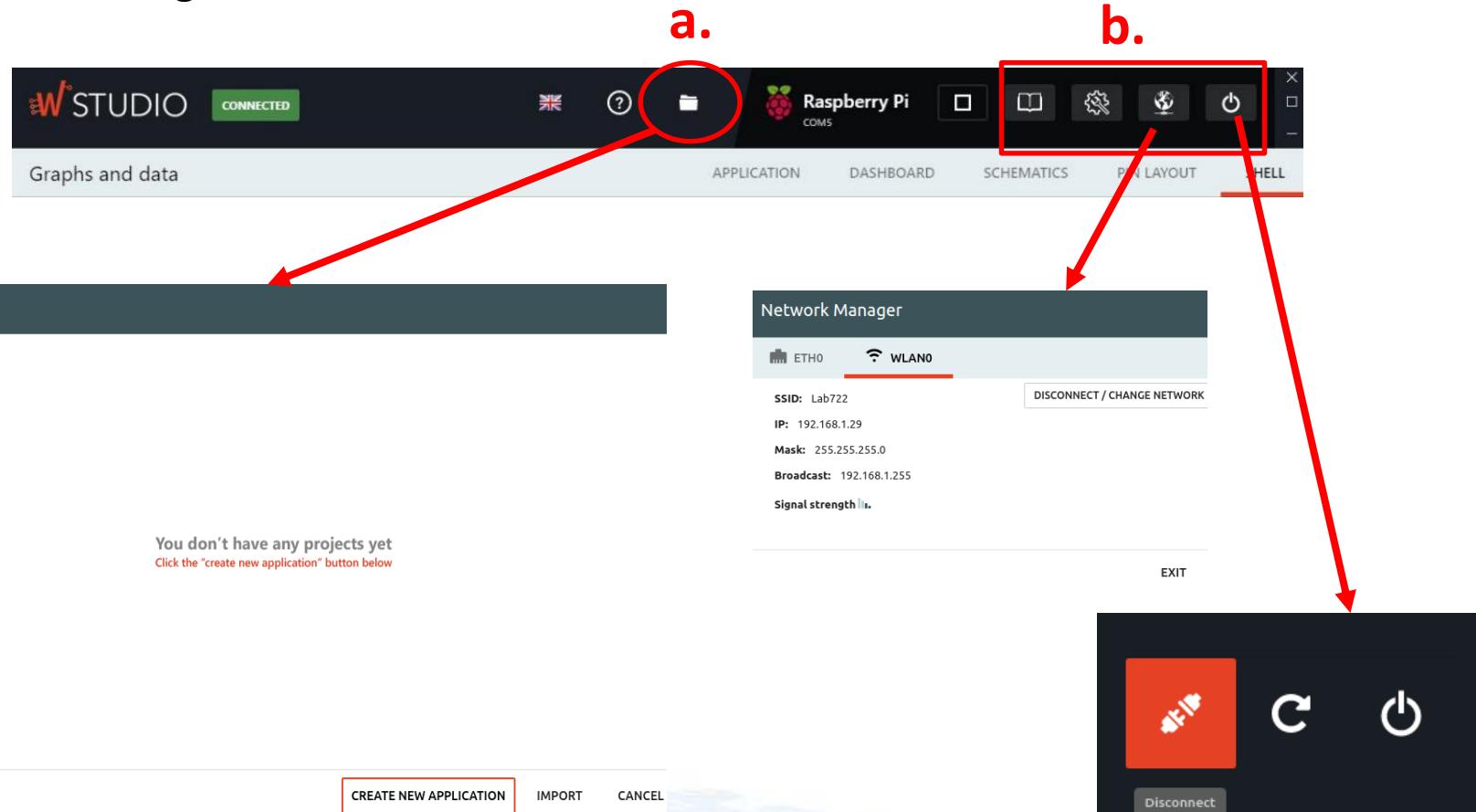
Shell

- Direct shell for advanced users (ssh terminal).



Wyliodrin Studio GUI (8)

- a. Project Manager
- b. Board Manager



CHECKPOINT 1!



RASPBERRY PI AND ARDUINO BASICS

Raspberry Pi (1)

- The Raspberry Pi is a credit-card-sized computer with integrated Ethernet, Wi-Fi, Bluetooth 4, HDMI, USB, and audio.
- Initially made in Taiwan and China, but now all are manufactured in the UK.
- Capable of hosting an OS (Raspbian).
- Multipurpose: Office, Internet, games, etc.
- GPIO (control sensors).



Raspberry Pi (2)

- Raspberry Pi can be directly used in electronics projects because it has a set of **general-purpose input/output (GPIO)** pins.
- These GPIO pins can be accessed for controlling sensors (temperature, light, motion, proximity, etc.) and actuators (LEDs, motors, relays, etc.).
- All of these GPIO pins are digital IO. In order to read/write analog inputs, an external ADC (analog-to-digital converter) can be used.

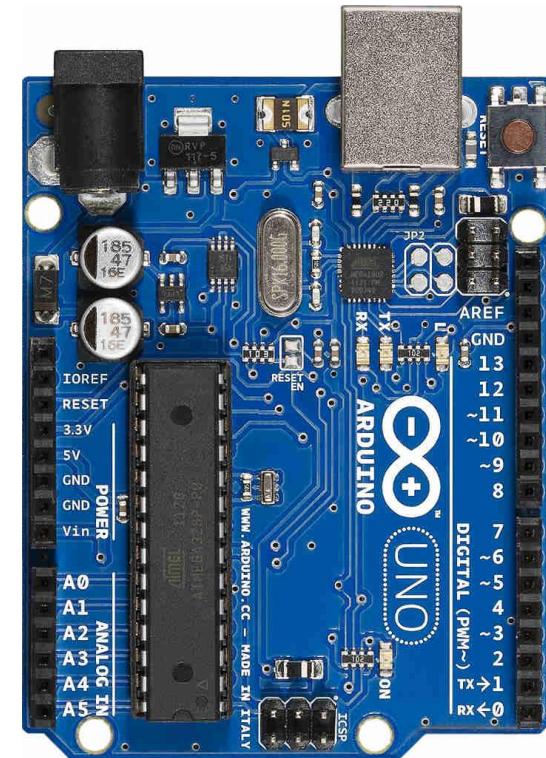
Raspberry Pi 3 Model B (J8 Header)	
GPIO#	NAME
8	3.3 VDC Power
8	GPIO 8 SDA1 (I2C)
9	GPIO 9 SCL1 (I2C)
7	GPIO 7 GPCLK0
	Ground
0	GPIO 0
2	GPIO 2
3	GPIO 3
	3.3 VDC Power
12	GPIO 12 MOSI (SPI)
13	GPIO 13 MISO (SPI)
14	GPIO 14 SCLK (SPI)
	Ground
30	SDA0 (I2C ID EEPROM)
21	GPIO 21 GPCLK1
22	GPIO 22 GPCLK2
23	GPIO 23 PWM1
24	GPIO 24 PCM_FS/PWM1
25	GPIO 25
	Ground
2	5.0 VDC Power
4	5.0 VDC Power
	Ground
6	GPIO 15 TxD (UART)
8	GPIO 16 RxD (UART)
10	PCM_CLK/PWM0
12	GPIO 1
14	Ground
16	GPIO 4
18	GPIO 5
20	Ground
22	GPIO 6
24	GPIO 10 CE0 (SPI)
26	GPIO 11 CE1 (SPI)
28	SCL0 (I2C ID EEPROM)
30	Ground
32	GPIO 26 PWM0
34	Ground
36	GPIO 27
38	GPIO 28 PCM_DIN
40	GPIO 29 PCM_DOUT

Attention! The GPIO pin numbering used in this diagram is intended for use with WiringPi / Pi4J. This pin numbering is not the raw Broadcom GPIO pin numbers.

<http://www.pi4j.com>

Arduino (1)

- Arduino is an open-source electronics platform.
- Consists of both a physical programmable circuit board (microcontroller) and an IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.
- It has analog and digital GPIO.
- Its functionality can be extended via Arduino Shields.



Arduino (2)

Digital Pins

Digital pins are GPIO.

Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (with a value of HIGH or LOW, respectively).

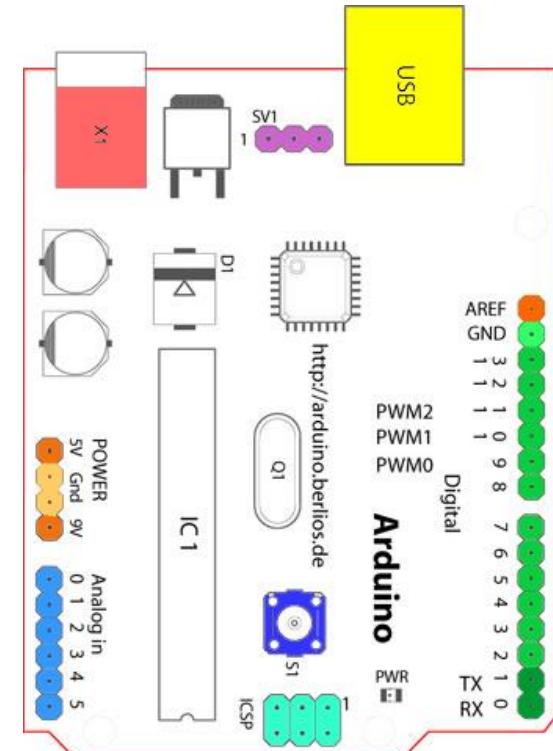
The maximum current per pin is 40 mA.

Analog Pins

Analog pins are input.

Support 10-bit analog-to-digital conversion (ADC) using the analogRead() function.

Most of the analog inputs can also be used as digital pins (depends on the model).



Arduino (3)

Why use Arduino when you have a Pi?

- The two platforms are complementary.
- There are quite a few situations where you might want to put the Arduino and Pi together:
 - To use the large number of libraries and sharable examples for the Arduino.
 - To supplement an Arduino project with more processing power.
 - When you're dealing with 5V logic levels. The Pi operates at 3.3V, and its pins are not tolerant of 5V. The Arduino can act as a "translator" between the two.

How Arduino and Raspberry Communicate?

- Installing "Firmata" is one of the solutions.
- Firmata is a generic protocol for communicating with microcontrollers from software on a host computer.
- It is intended to work with any host computer software package.

Arduino : Installing Firmata (1)

1. Download and Install the Arduino IDE Application in your computer from <https://www.arduino.cc/en/Main/Software>.



Download the Arduino Software

This screenshot shows the "Download" section of the Arduino website. On the left, there is a large circular icon with the Arduino logo. Next to it, the text "ARDUINO 1.6.4" is displayed, followed by a detailed description of the software. On the right, there is a teal-colored sidebar with download links for different operating systems: "Windows Installer", "Windows ZIP file for non admin install", "Mac OS X 10.7 Lion or newer", "Linux 32 bits", and "Linux 64 bits". At the bottom of the sidebar, there is a link to "Release Notes".

ARDUINO 1.6.4

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer

Windows ZIP file for non admin install

Mac OS X 10.7 Lion or newer

Linux 32 bits

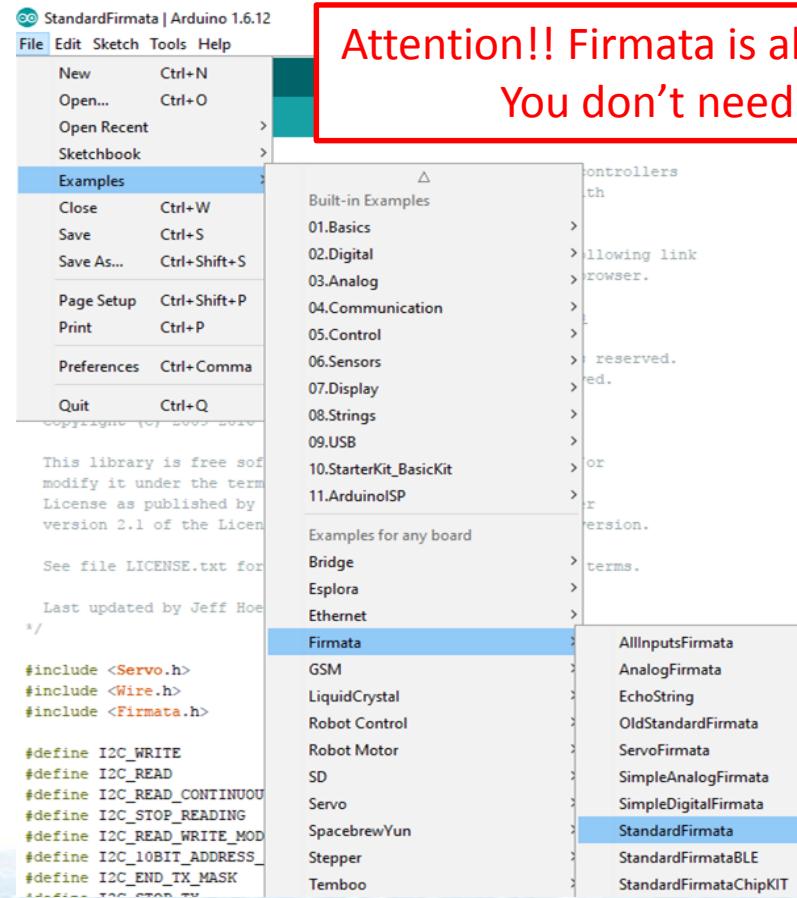
Linux 64 bits

[Release Notes](#)

Attention!! Firmata is already installed in your Arduino.
You don't need to execute these steps.

Arduino : Installing Firmata (2)

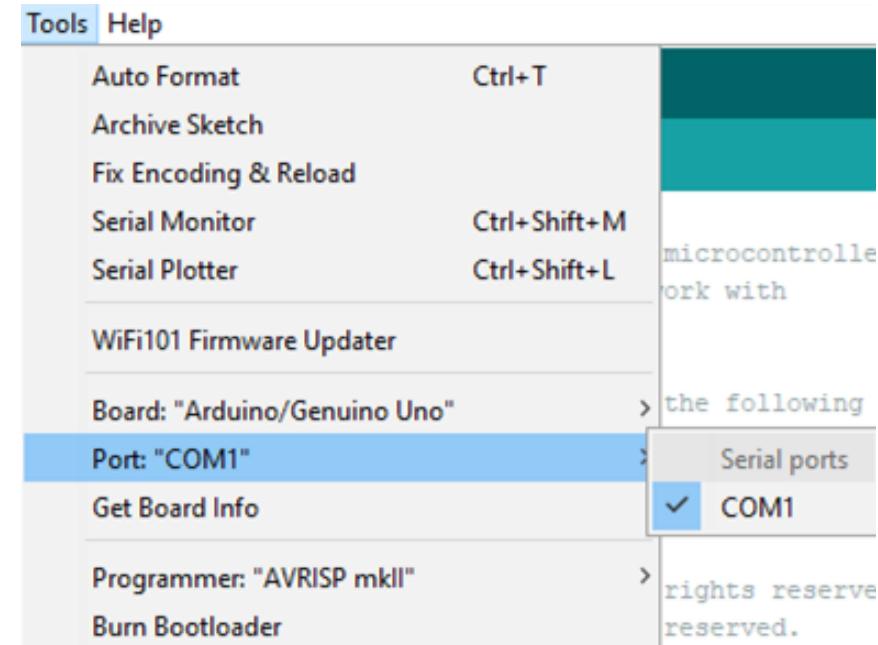
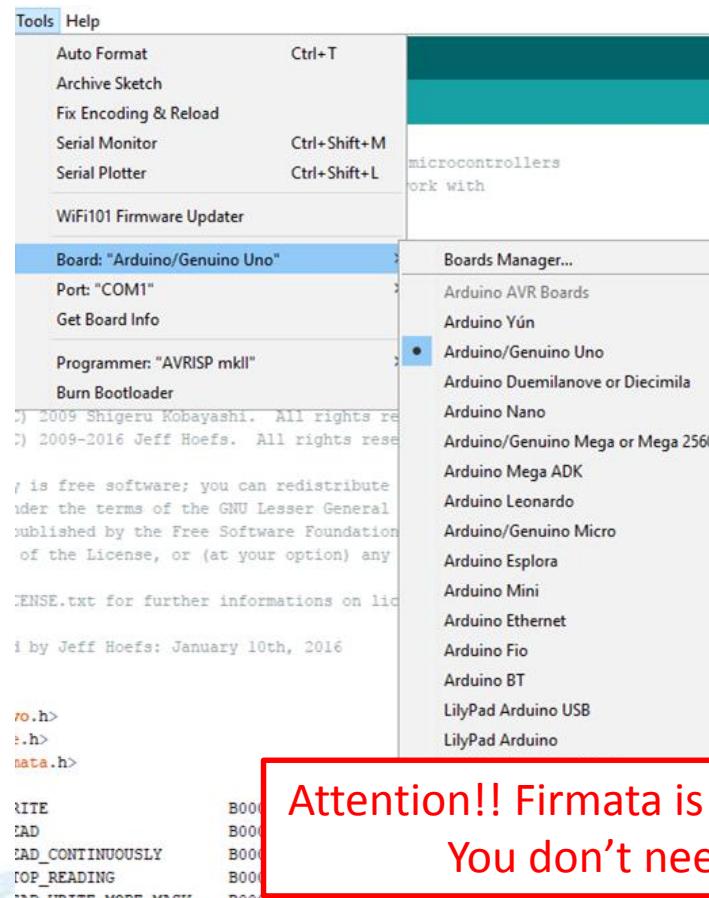
2. Open Arduino IDE application and plug the Arduino board into a your computer.
3. Open File > Examples > Firmata > StandardFirmata.



Attention!! Firmata is already installed in your Arduino.
You don't need to execute these steps.

Arduino : Installing Firmata (3)

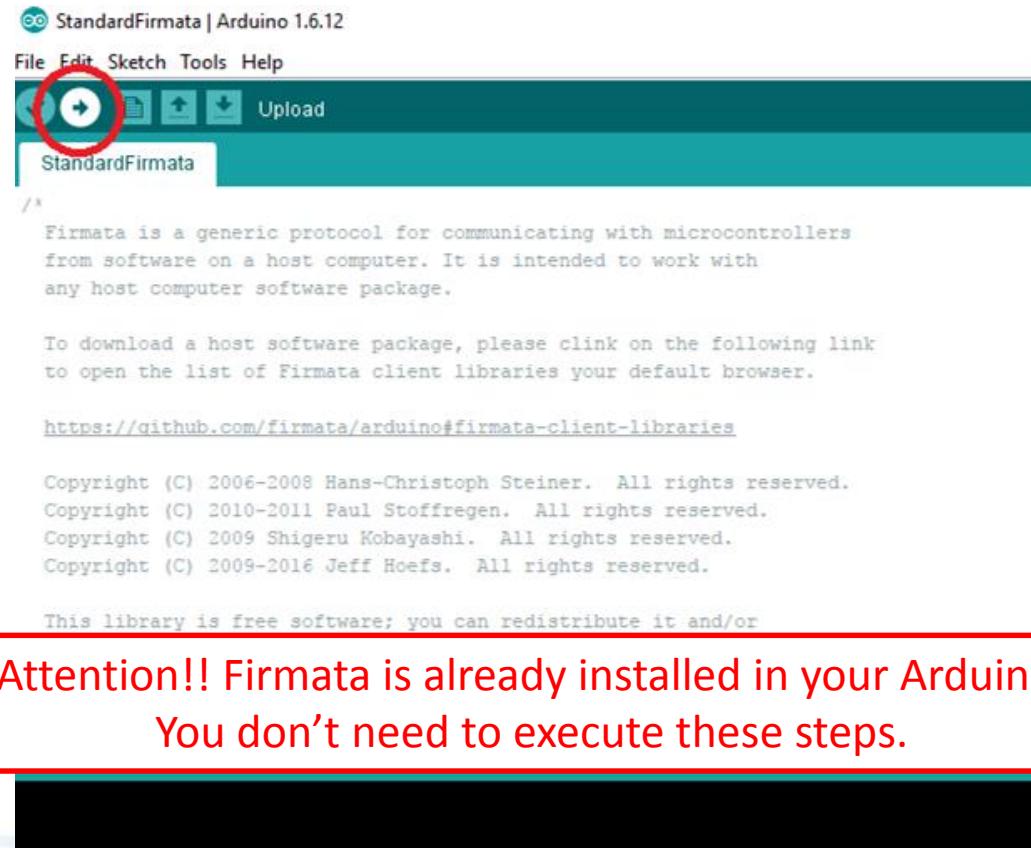
4. Select your Arduino board model in Tools > Board: "Your Arduino Model".
5. Select the port which your board is connected in Tools > Port: "Your Arduino Port".



Attention!! Firmata is already installed in your Arduino.
You don't need to execute these steps.

Arduino : Installing Firmata (4)

- Upload “StandartFirmata” into your board by pressing highlighted button or simply choose Sketch > Upload.





HELLO WORLD APPLICATION

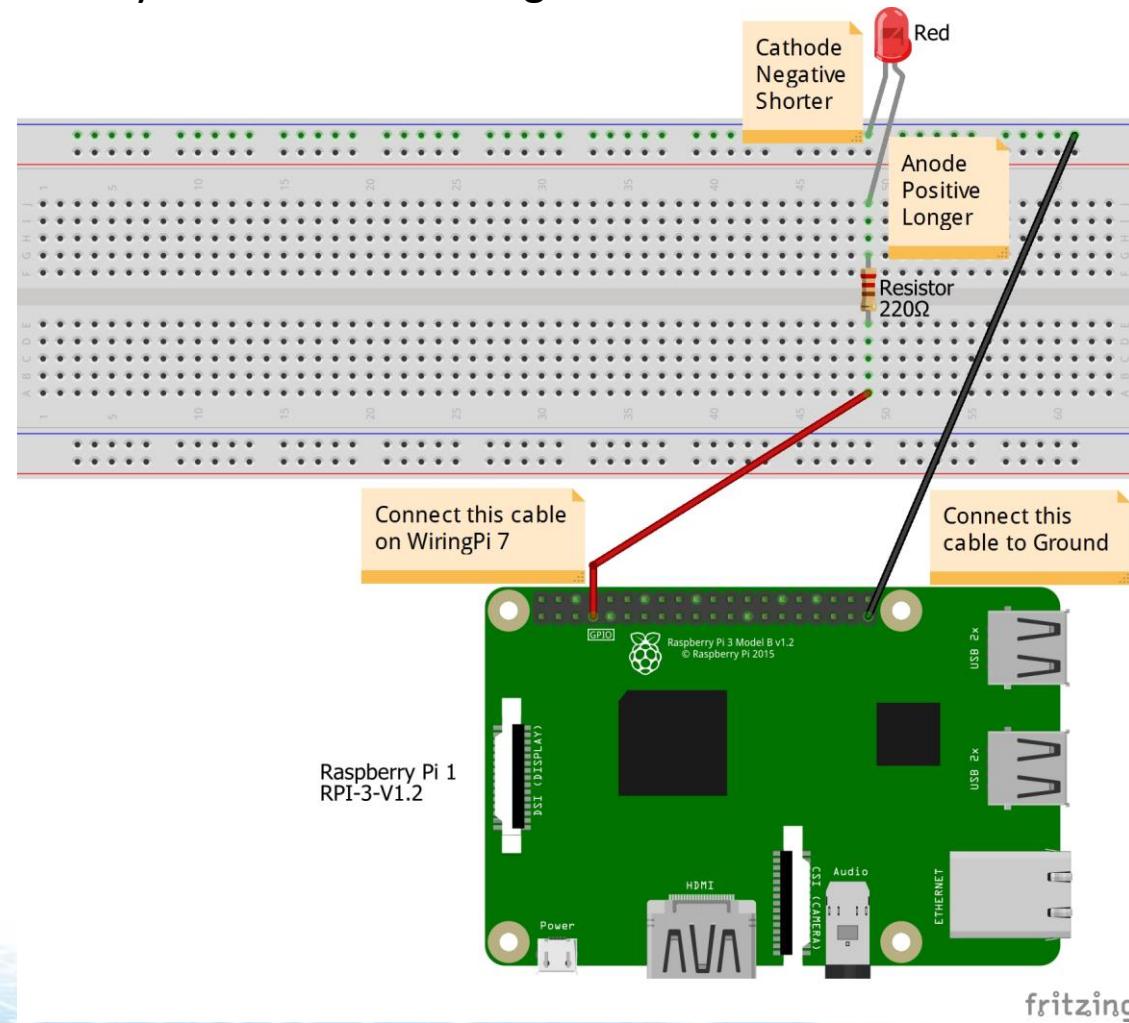
Hello World Application (1)

1. Prepare your Raspberry Pi and your board according to the schematics below.

Goal: Turn ON/OFF LED

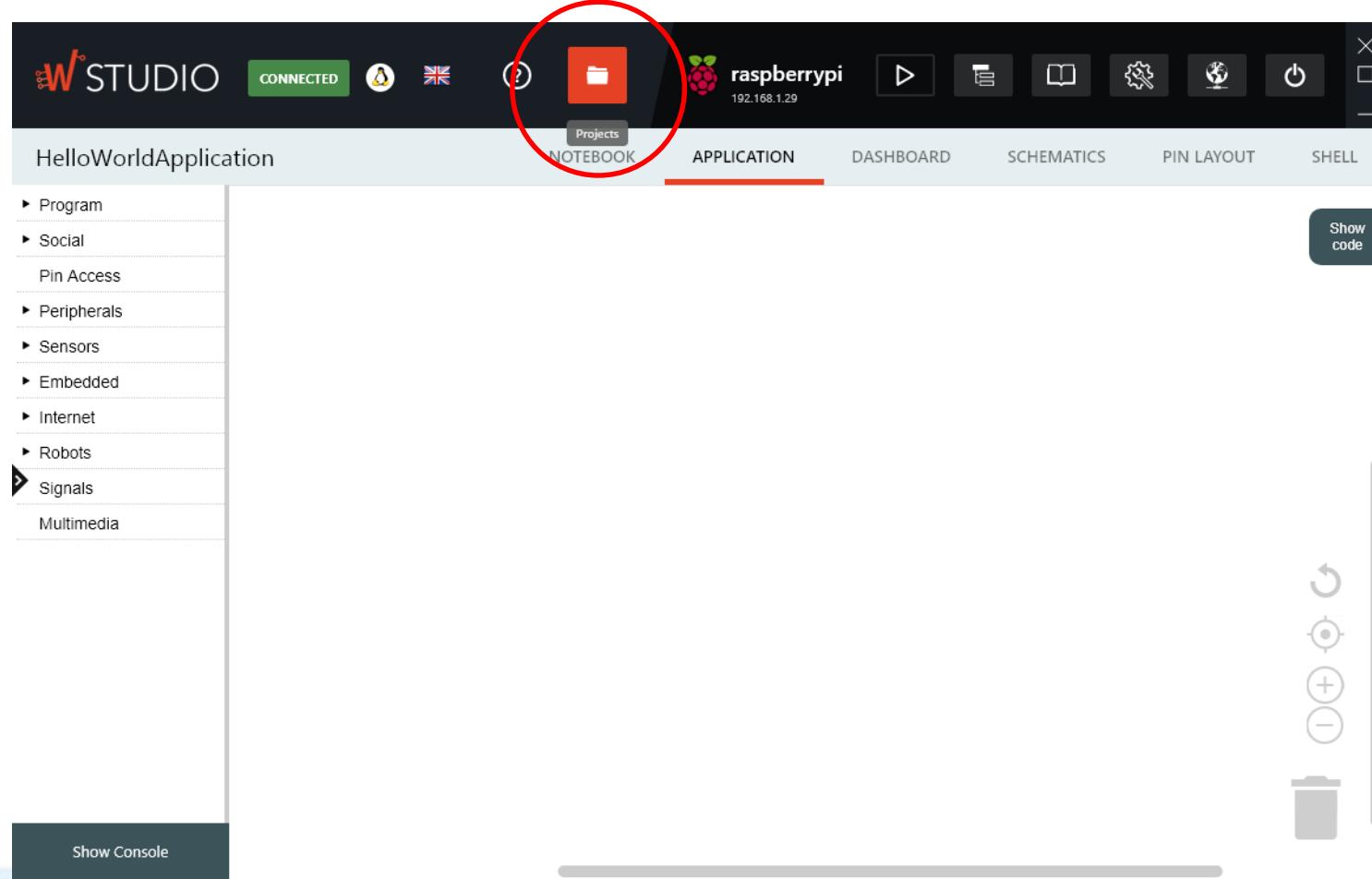
Components:

- 1 Raspberry Pi.
- 1 LED.
- 1 Resistor 220.
- Cables.



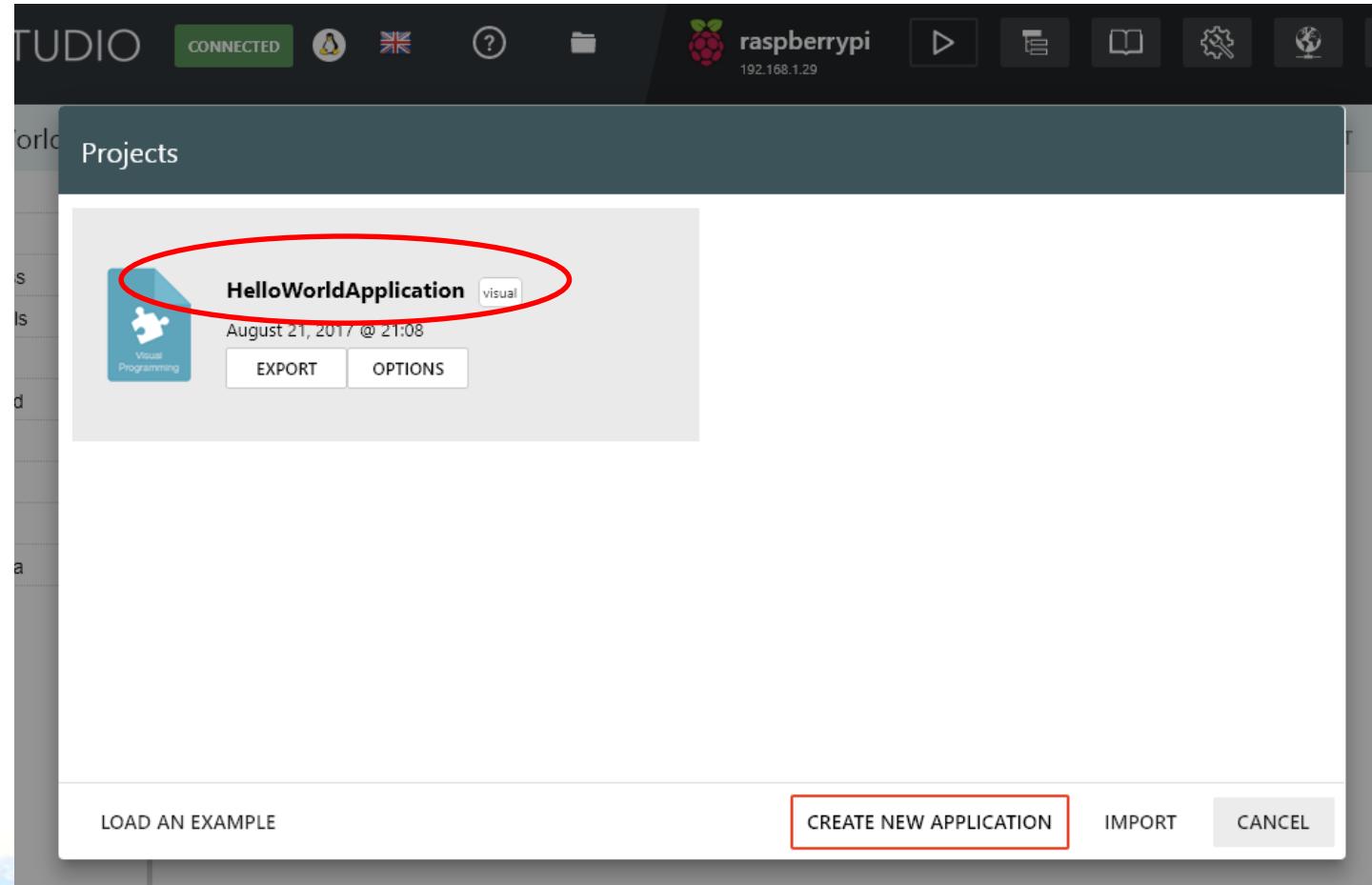
Hello World Application (2)

2. Click on “Projects”.



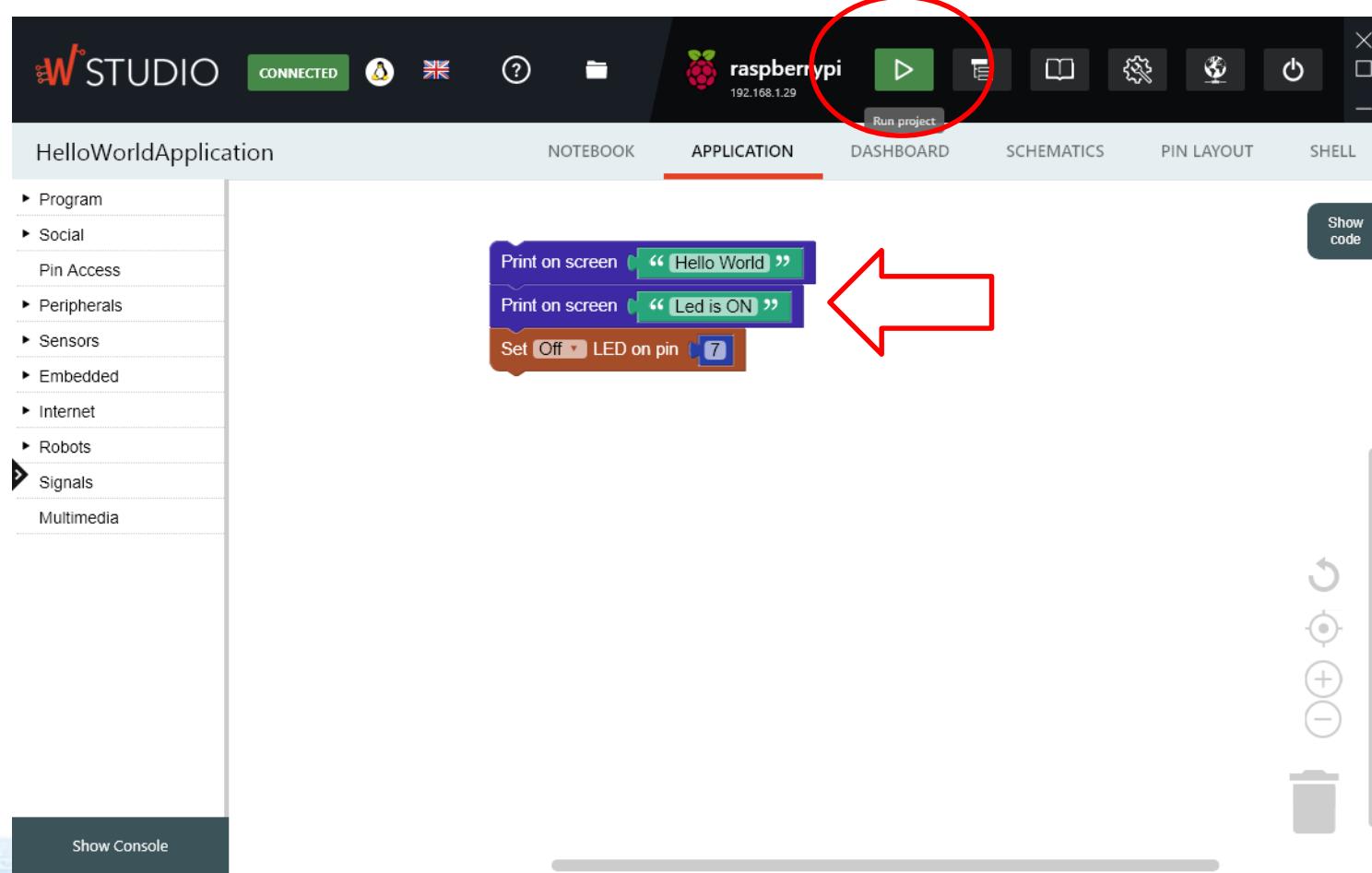
Hello World Application (3)

3. Click on “HelloWorldApplication”.



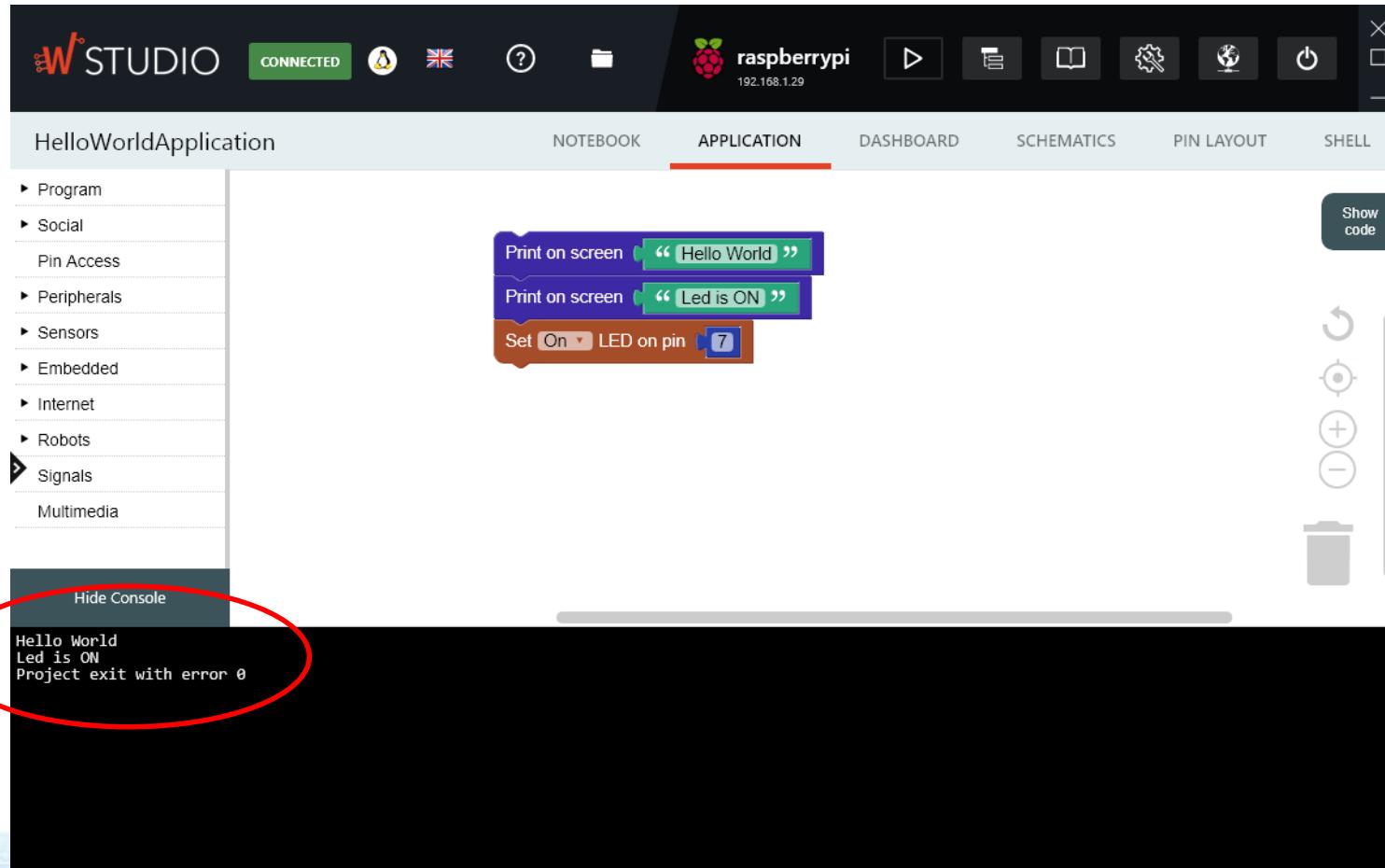
Hello World Application (4)

4. In the “Application” tab you will use a few blocks.
5. Click on “Run Project”.



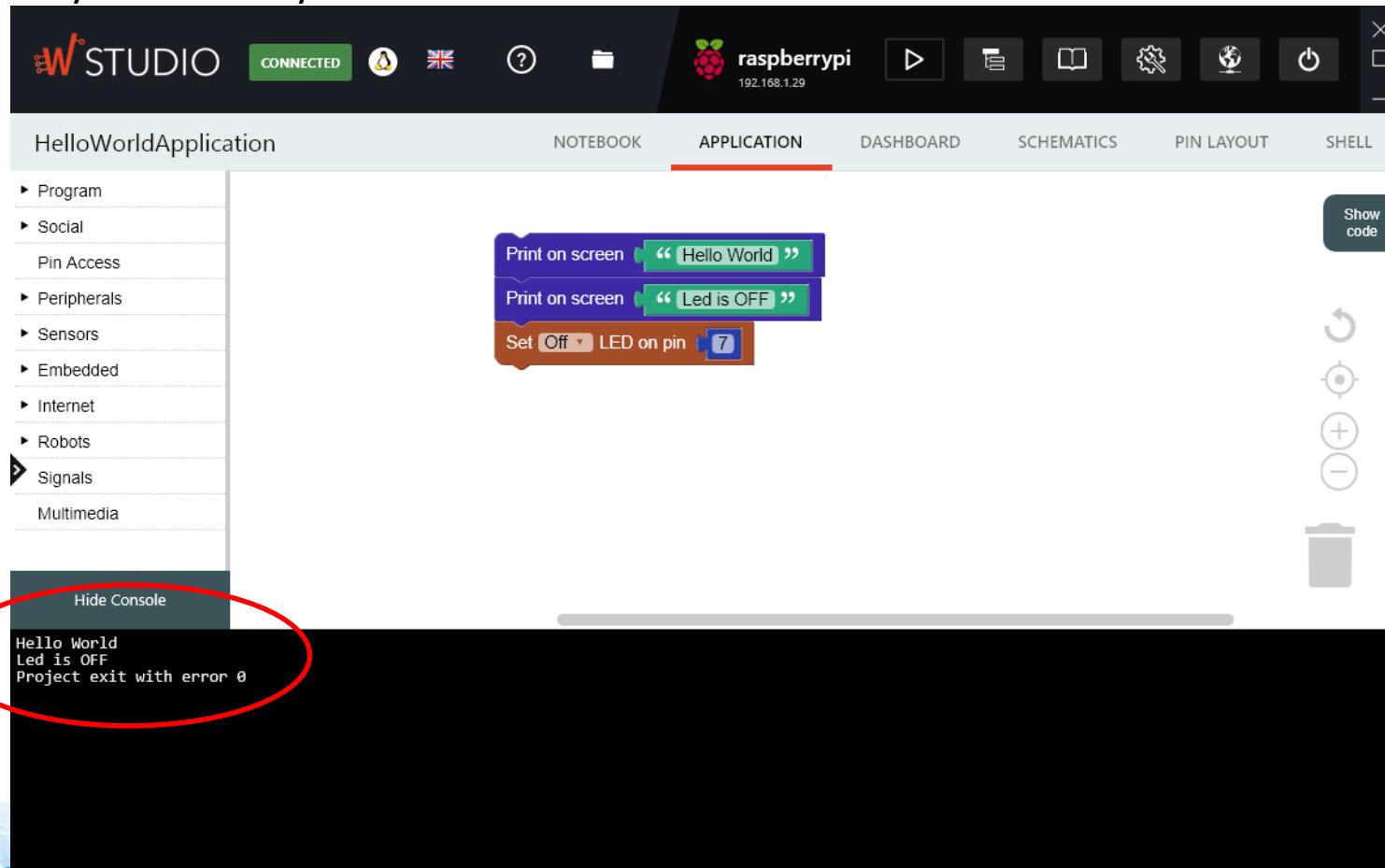
Hello World Application (5)

6. Notice the output in the console.
7. Verify the LED in your board is turned ON.



Hello World Application (6)

8. Modify the program so that the LED is OFF.
9. Modify also the message on screen to “LED is OFF”.
10. Verify the LED in your board is turned OFF.



CHECKPOINT 2!



TRAFFIC LIGHT SIMULATOR APPLICATION

Goal

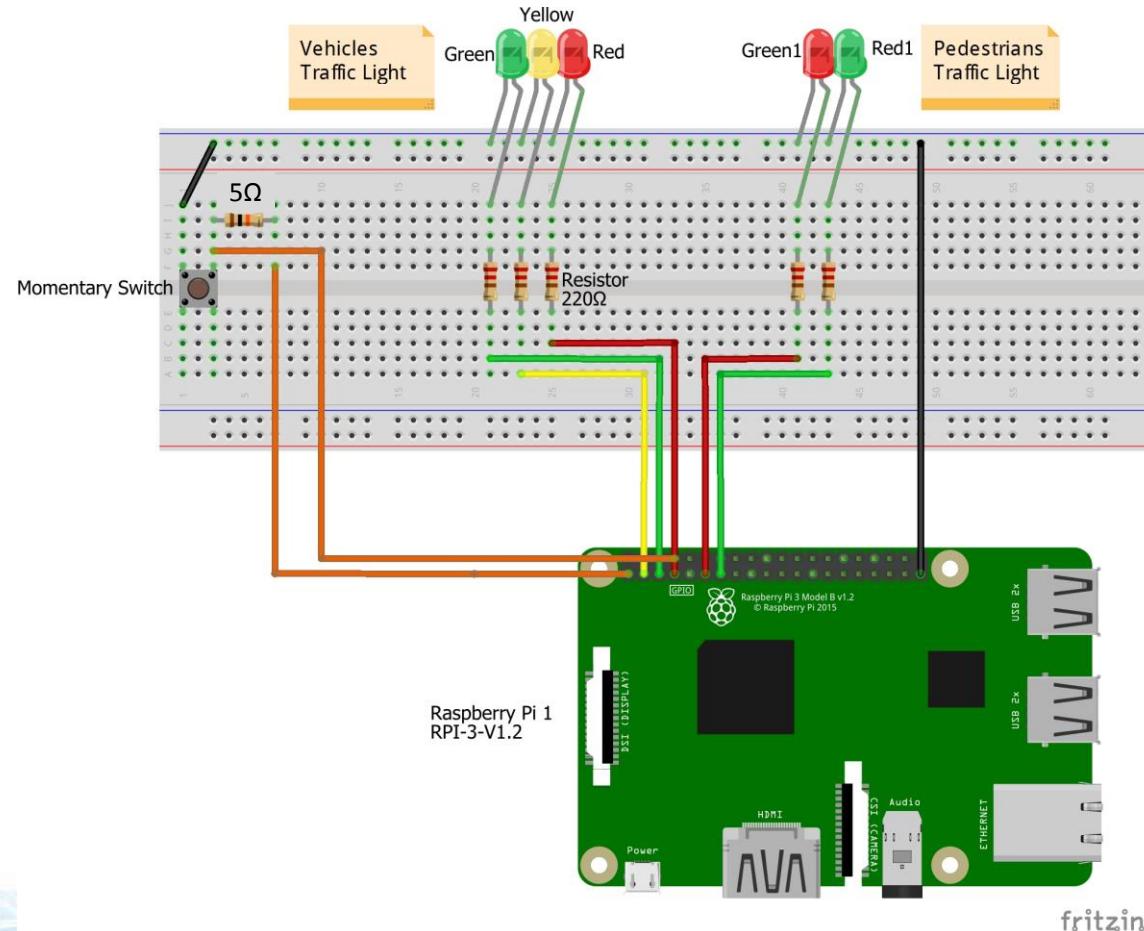
- Simulate a vehicles traffic light and a pedestrian traffic light:
 1. When the pedestrian pushes a button, the vehicles traffic light should change from green, to yellow, and finally red.
 2. The pedestrian traffic light should change from red to green at the same time as (1) finishes.
 3. The pedestrian traffic light should remain 30 seconds in green color before returning to red color, and at the same time the vehicles traffic light should return to green color.

List of Components

- 1 breadboard.
- 1 Raspberry Pi 3.
- 5 resistors $220\ \Omega$, 1 resistor 5Ω .
- 2 LED green, 2 LED red, 1 LED yellow.
- 1 momentary switch.
- 9 cables.

Schematics

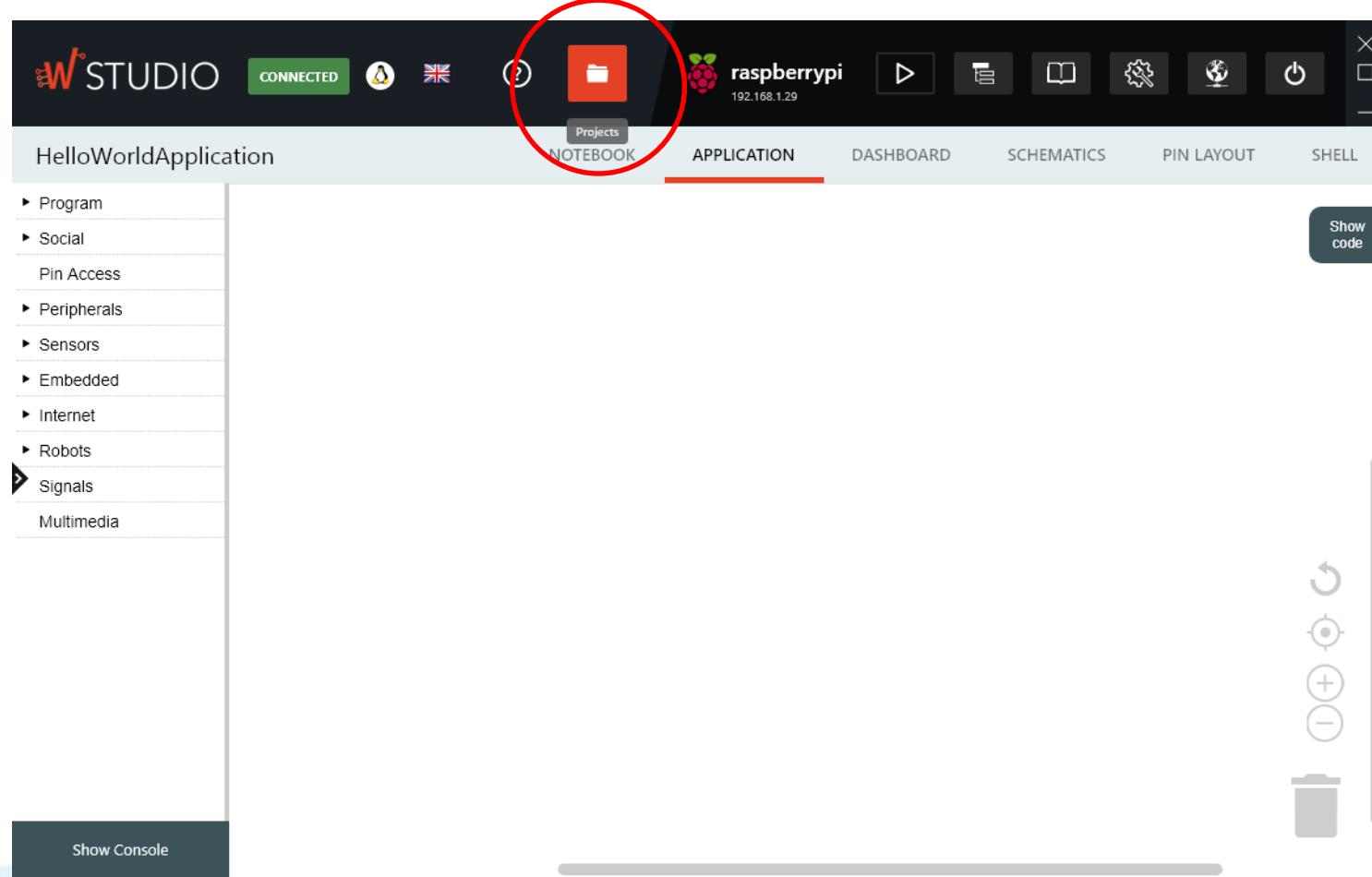
Connect all the components according to the schema.
Ask TA if you are not sure how to connect the components.



CHECKPOINT 3!

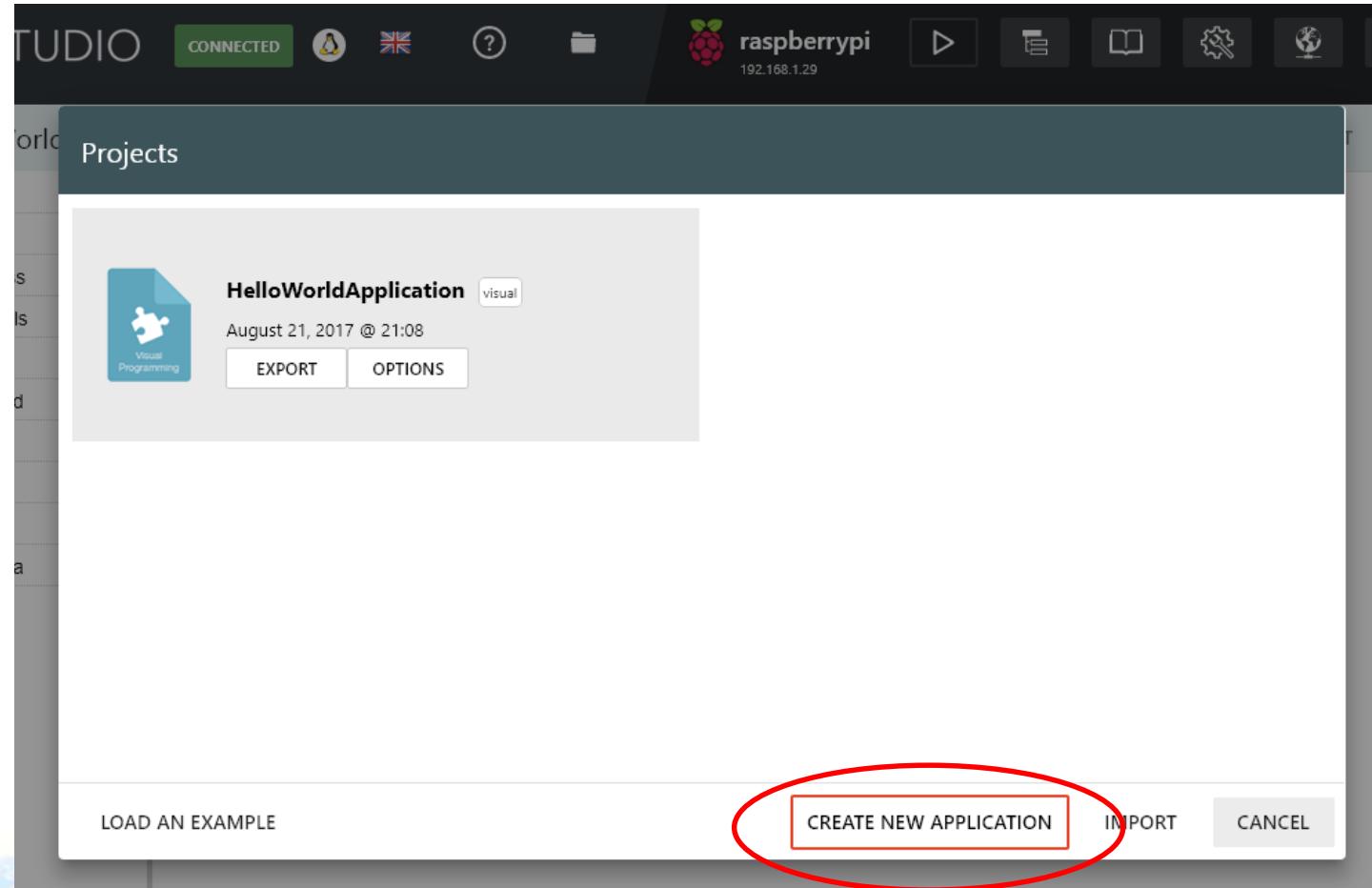
Traffic Light Application (1)

1. Click on “Projects”.



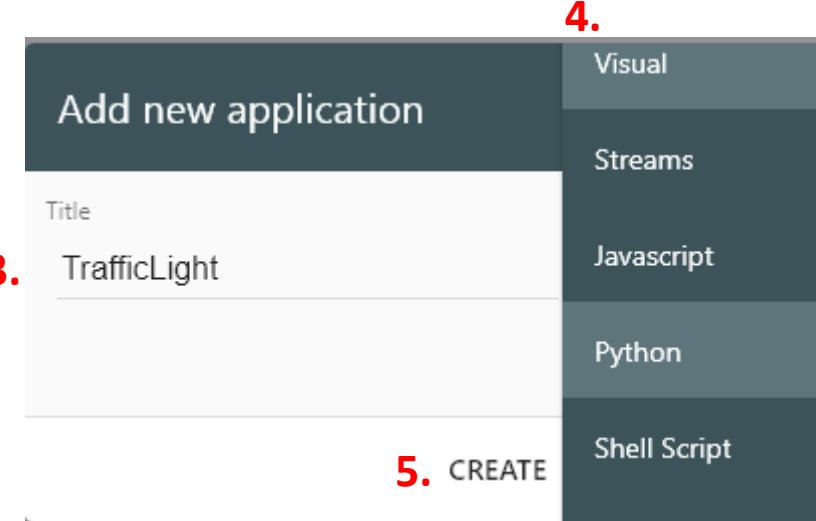
Traffic Light Application (2)

2. Click on “CREATE NEW APPLICATION”.



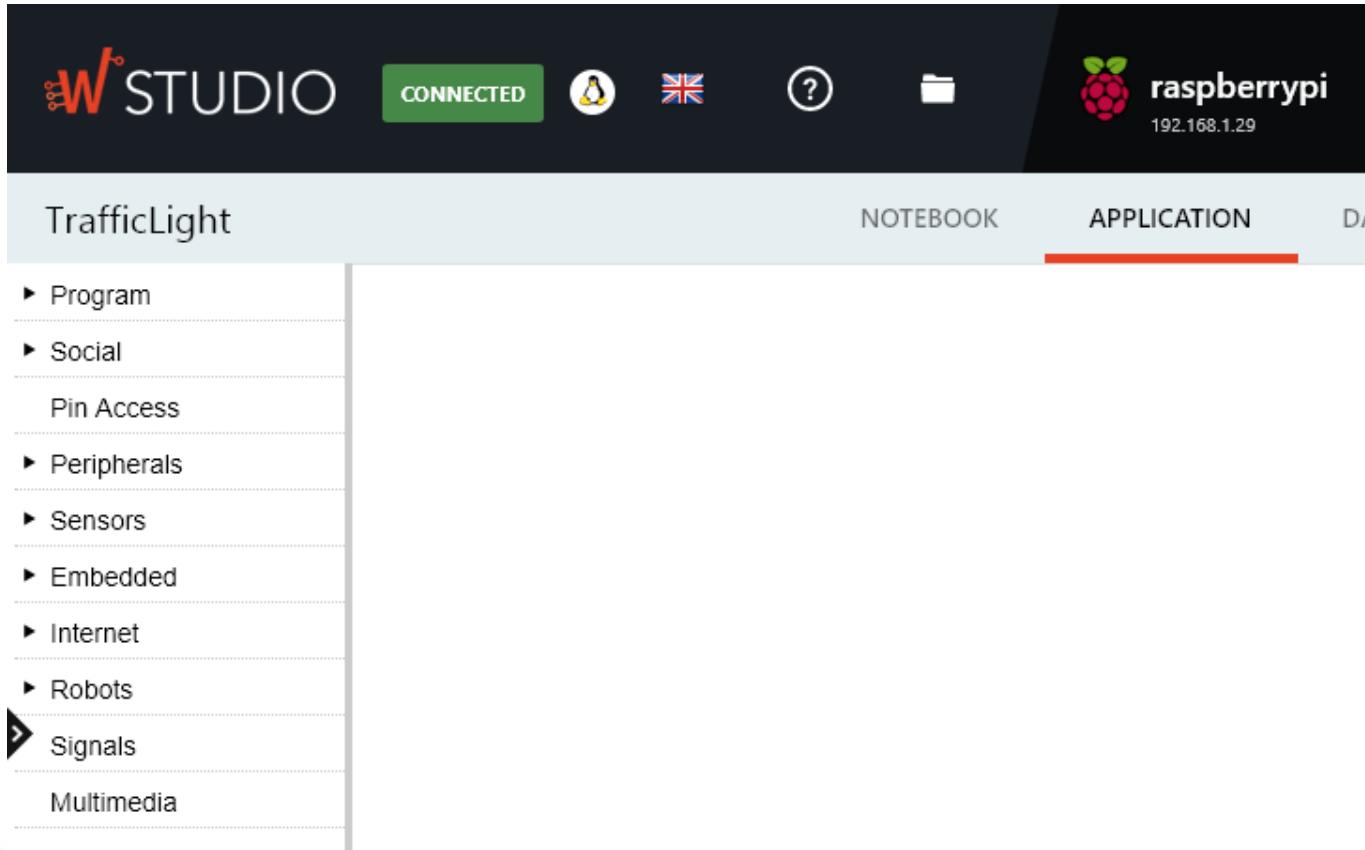
Traffic Light Application (3)

3. Input “TrafficLight” as the Title.
4. Select “Visual” as the Language.
5. Click on “Create”.



Coding with Blocks (1)

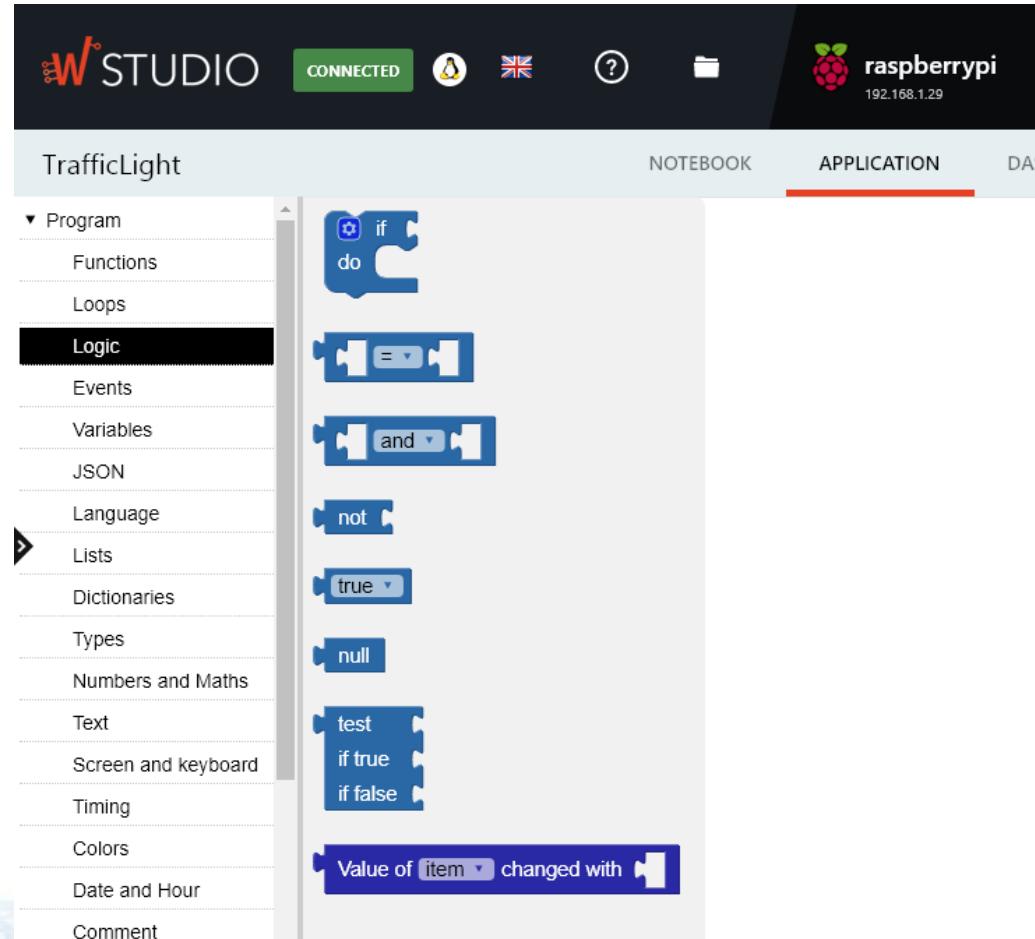
Visual Programming is a language based on [Google Blockly](#) and its purpose is learning. While writing your program in Visual Programming, you can see the Python output.



The screenshot shows the W-STUDIO application interface. At the top, there's a header bar with the W-STUDIO logo, a green "CONNECTED" button, icons for Linux, UK, help, and a folder, and a Raspberry Pi icon with the IP address 192.168.1.29. Below the header is a navigation bar with tabs: "TrafficLight" (selected), "NOTEBOOK", "APPLICATION" (underlined in red), and "DA". On the left, a sidebar lists categories with arrows: Program, Social, Pin Access, Peripherals, Sensors, Embedded, Internet, Robots, Signals (which is expanded to show sub-options like Analog Input, Digital Input, Pulse Width Modulation, and PWM Frequency), and Multimedia.

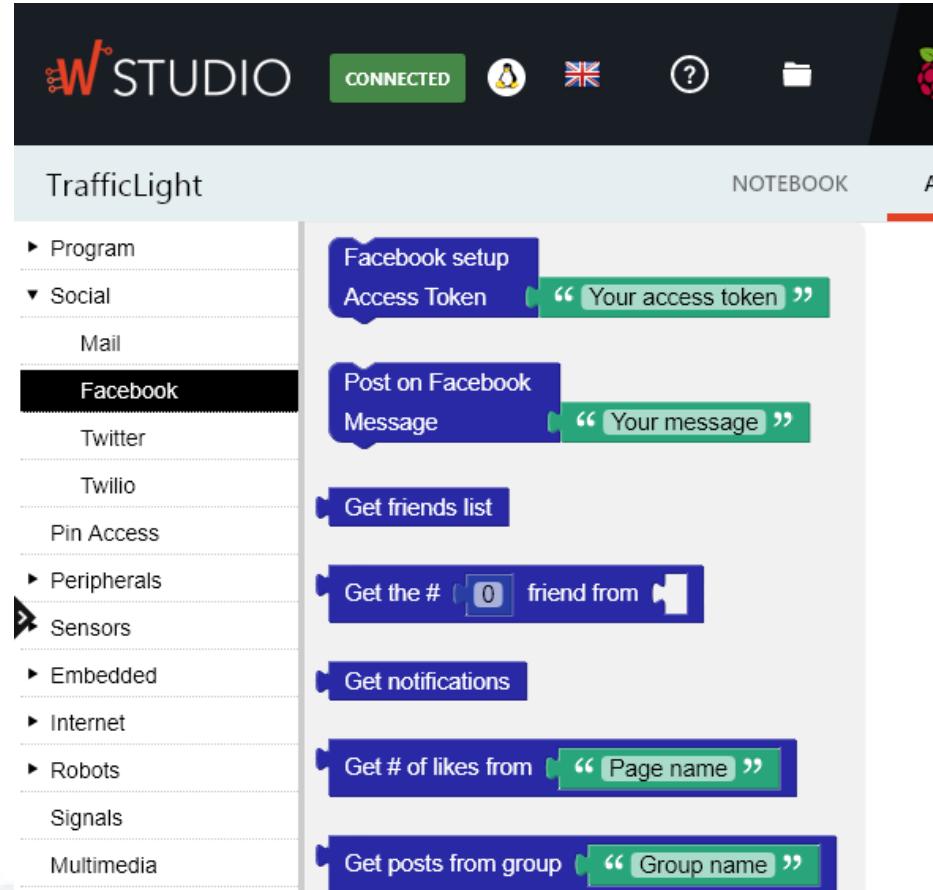
Coding with Blocks (2)

Program includes functions, loops, logic, variables, printing, timing, lists, math, etc.



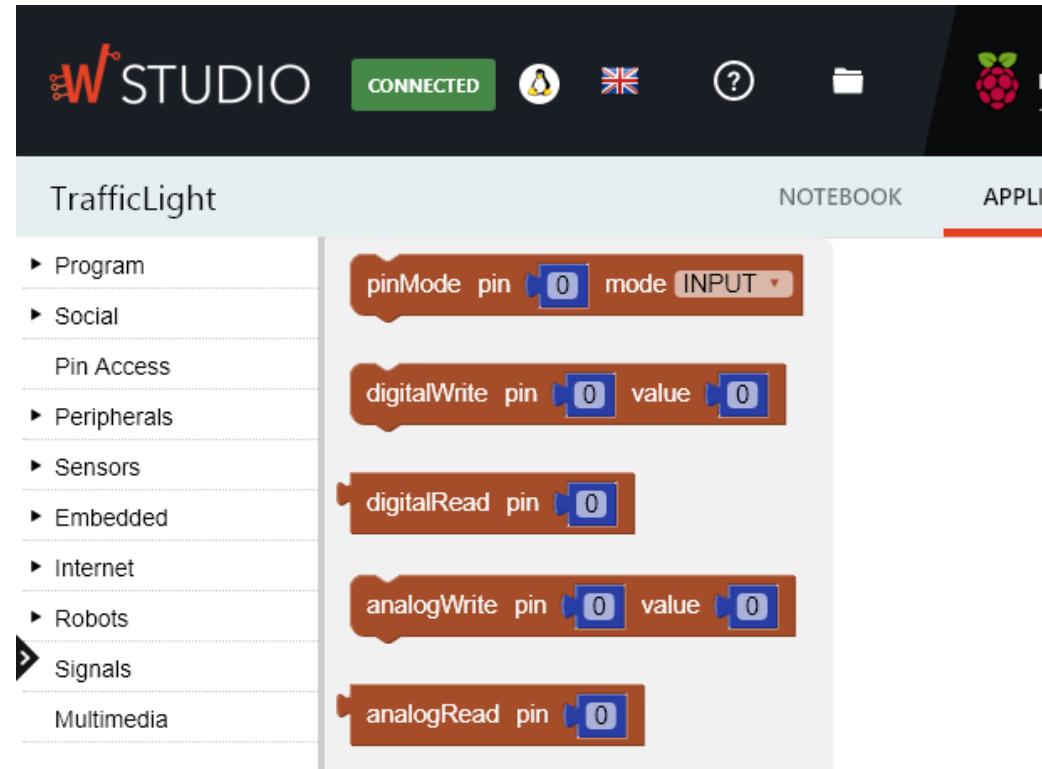
Coding with Blocks (3)

Social includes blocks for Email, Facebook, Twitter, and Twilio.



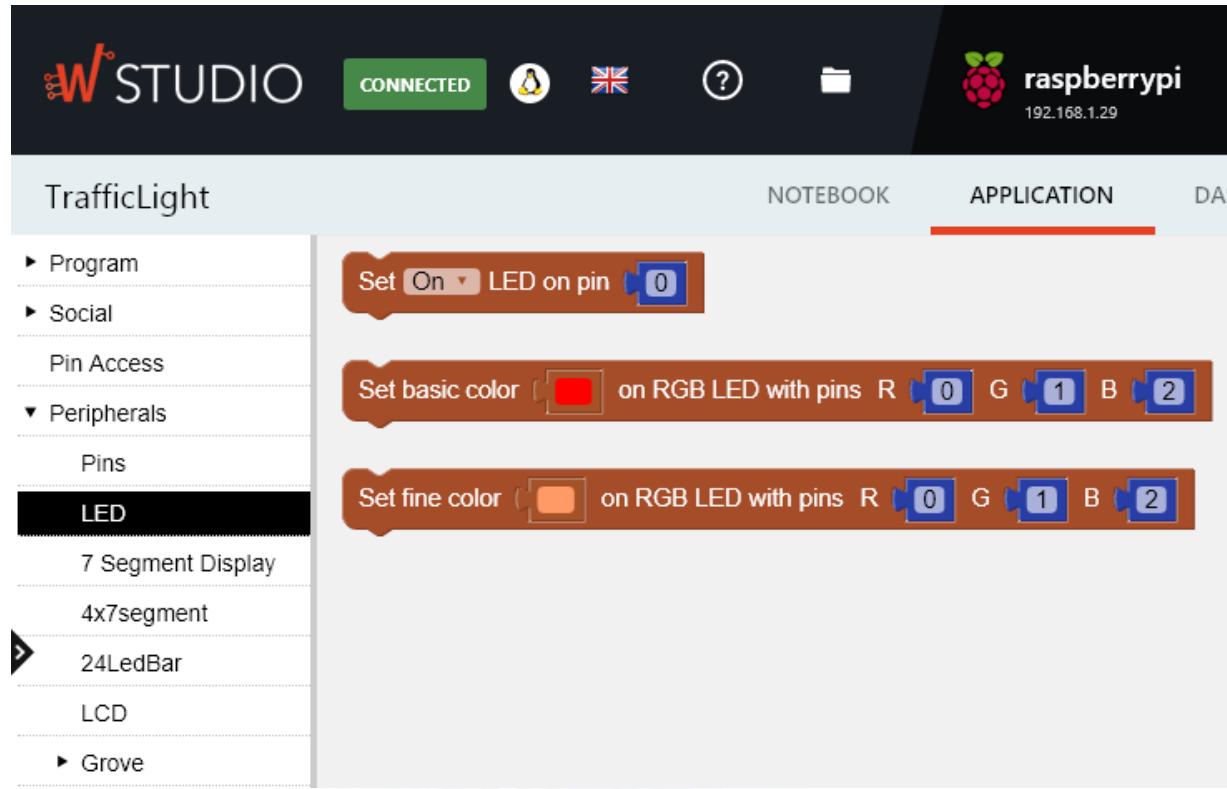
Coding with Blocks (4)

Pin Access includes blocks for setting the mode of boards' pin to Input or Output, and reading/writing Digital and Analog pins in the boards.



Coding with Blocks (5)

Peripherals provides blocks for accessing LEDs (turn ON/OFF) and Digital Displays.

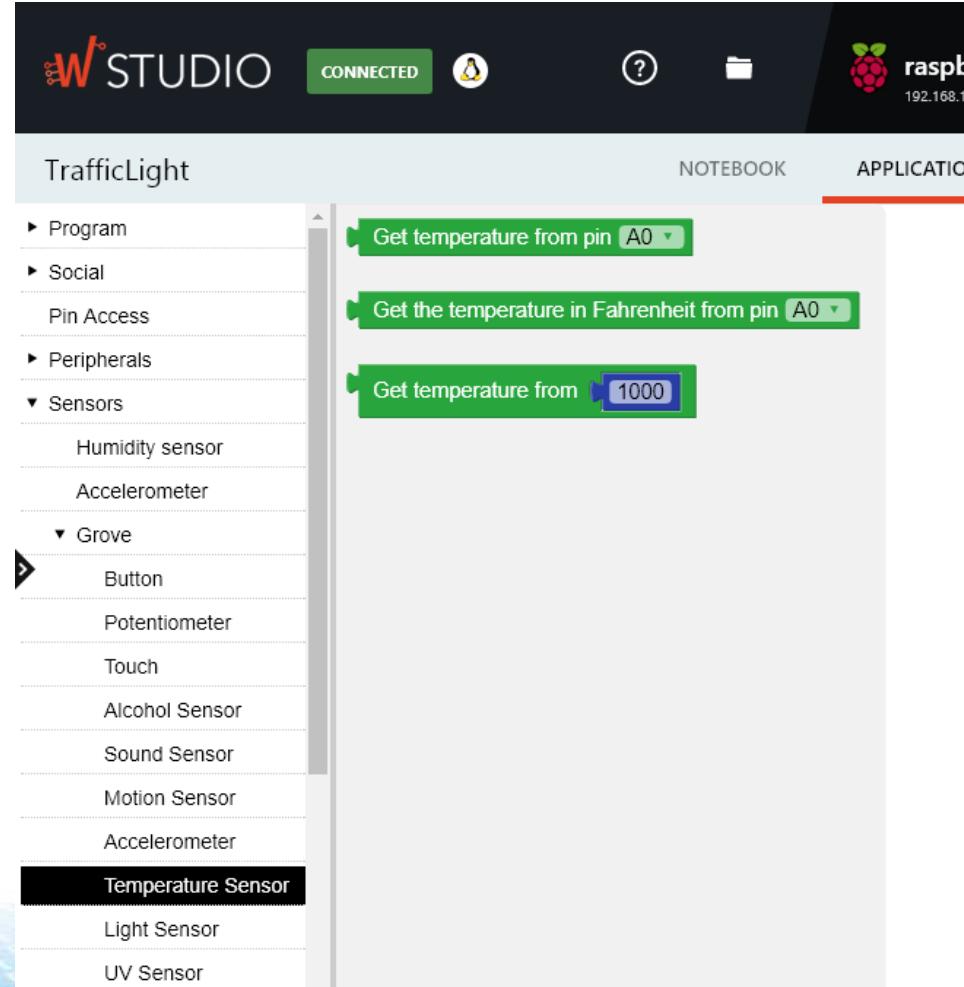


The screenshot shows the WSTUDIO interface connected to a raspberry pi (192.168.1.29). The application tab is selected. On the left, a sidebar menu under the Peripherals section has 'LED' selected. Three blocks are visible on the canvas:

- Set [On] LED on pin 0
- Set basic color [red] on RGB LED with pins R 0 G 1 B 2
- Set fine color [orange] on RGB LED with pins R 0 G 1 B 2

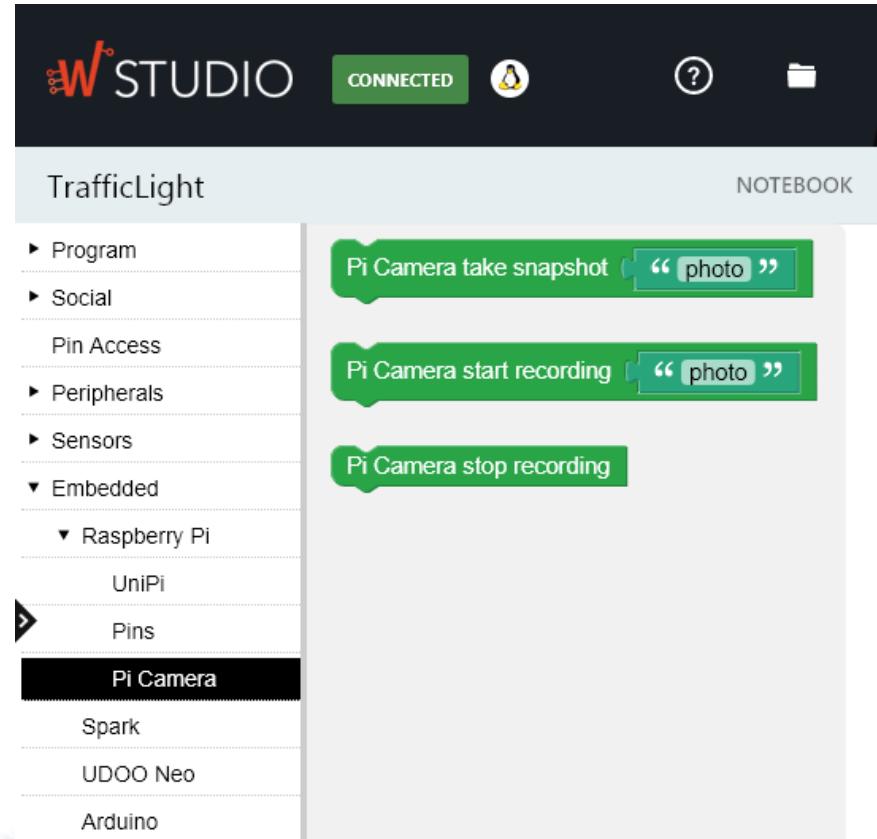
Coding with Blocks (6)

Sensors provides blocks for reading values from Arduino, Grove, and Raspberry Pi common sensors.



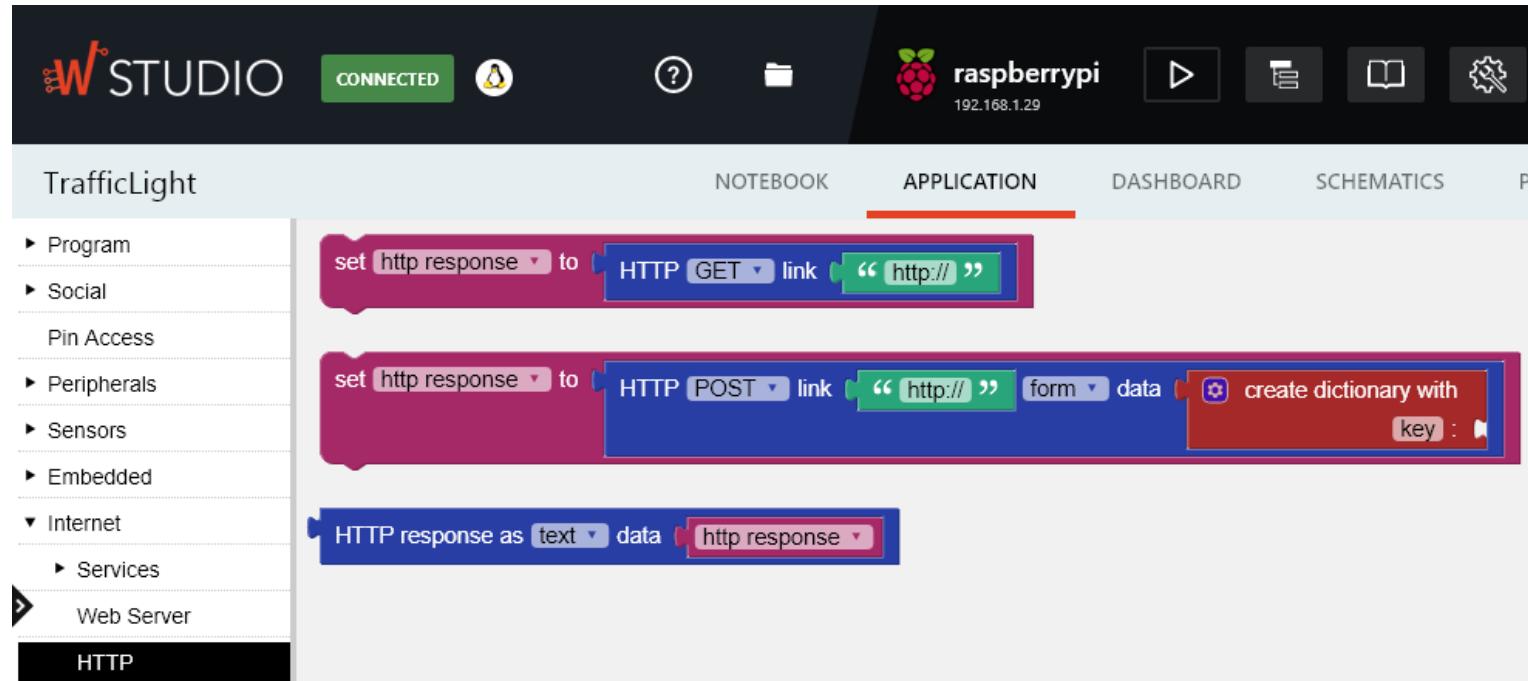
Coding with Blocks (7)

Embedded allows manipulating some actuators in UniPi, Pi Camera, Spark, UDOO Neo, and Arduino.



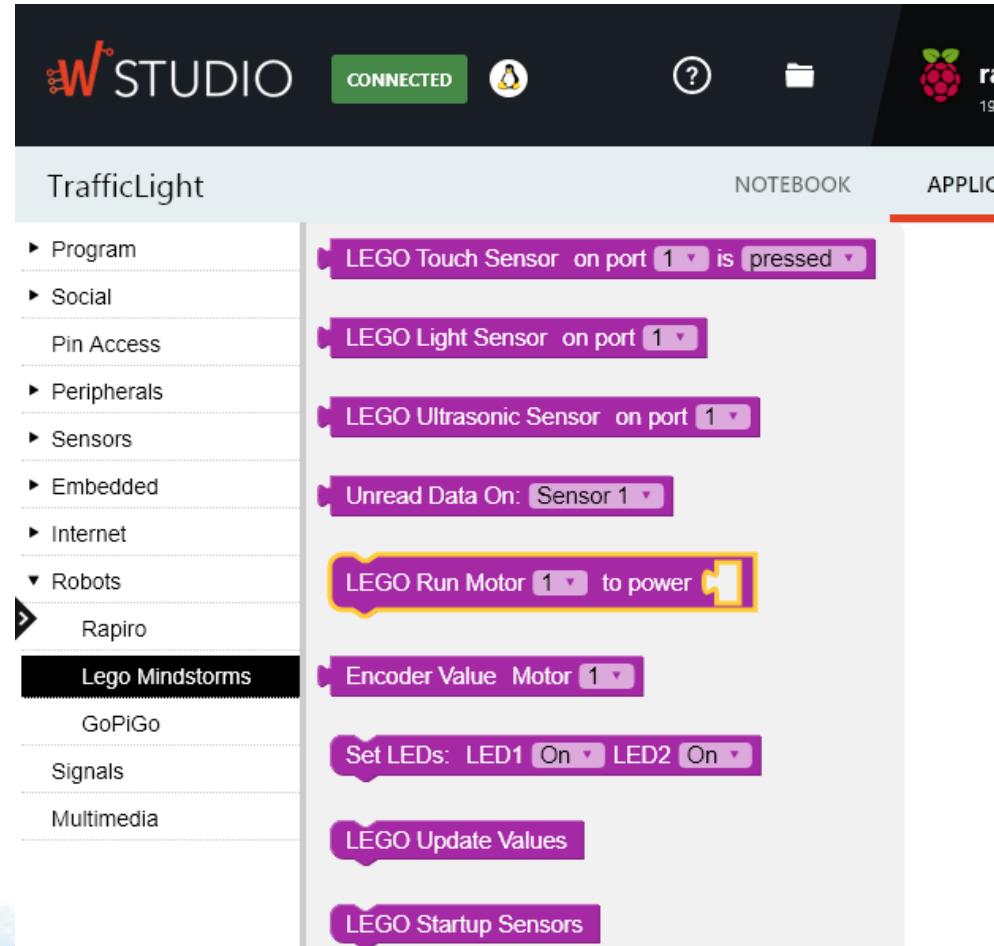
Coding with Blocks (8)

Internet contain blocks for HTTP RESTful communication and web services.



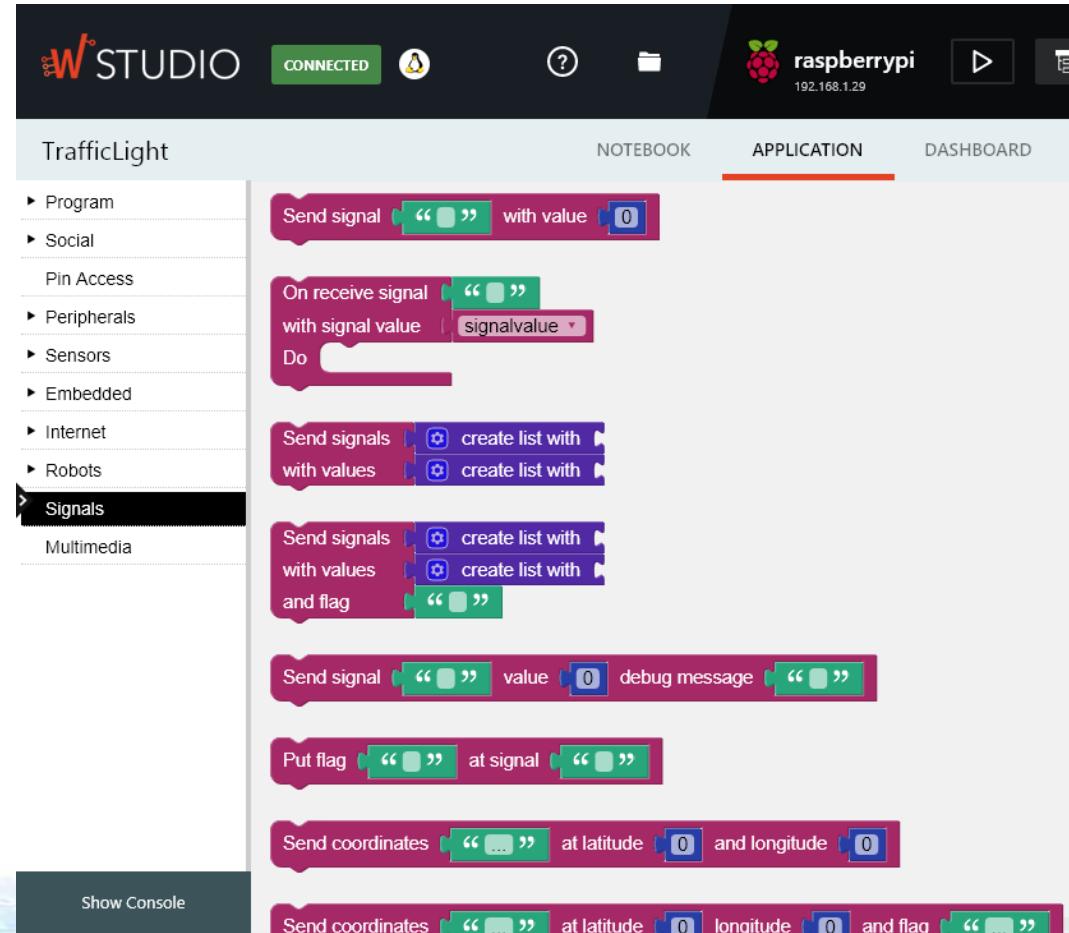
Coding with Blocks (9)

Robots contain blocks for interacting with Rapiro, Lego Mindstorms, and GoPiGo robots.



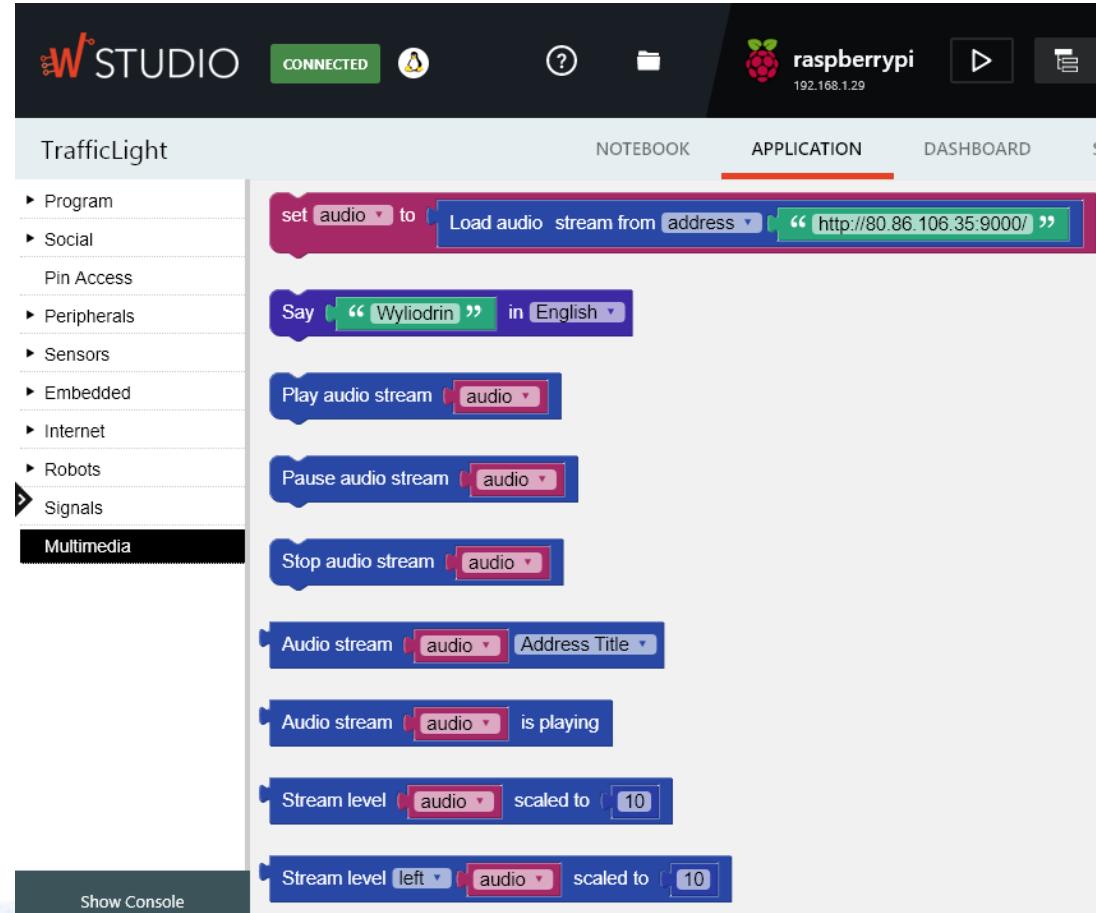
Coding with Blocks (10)

Signals can be utilized to create time-based or real-time output the sensor values. These output can be visualized later in the “Dashboard” tab.



Coding with Blocks (11)

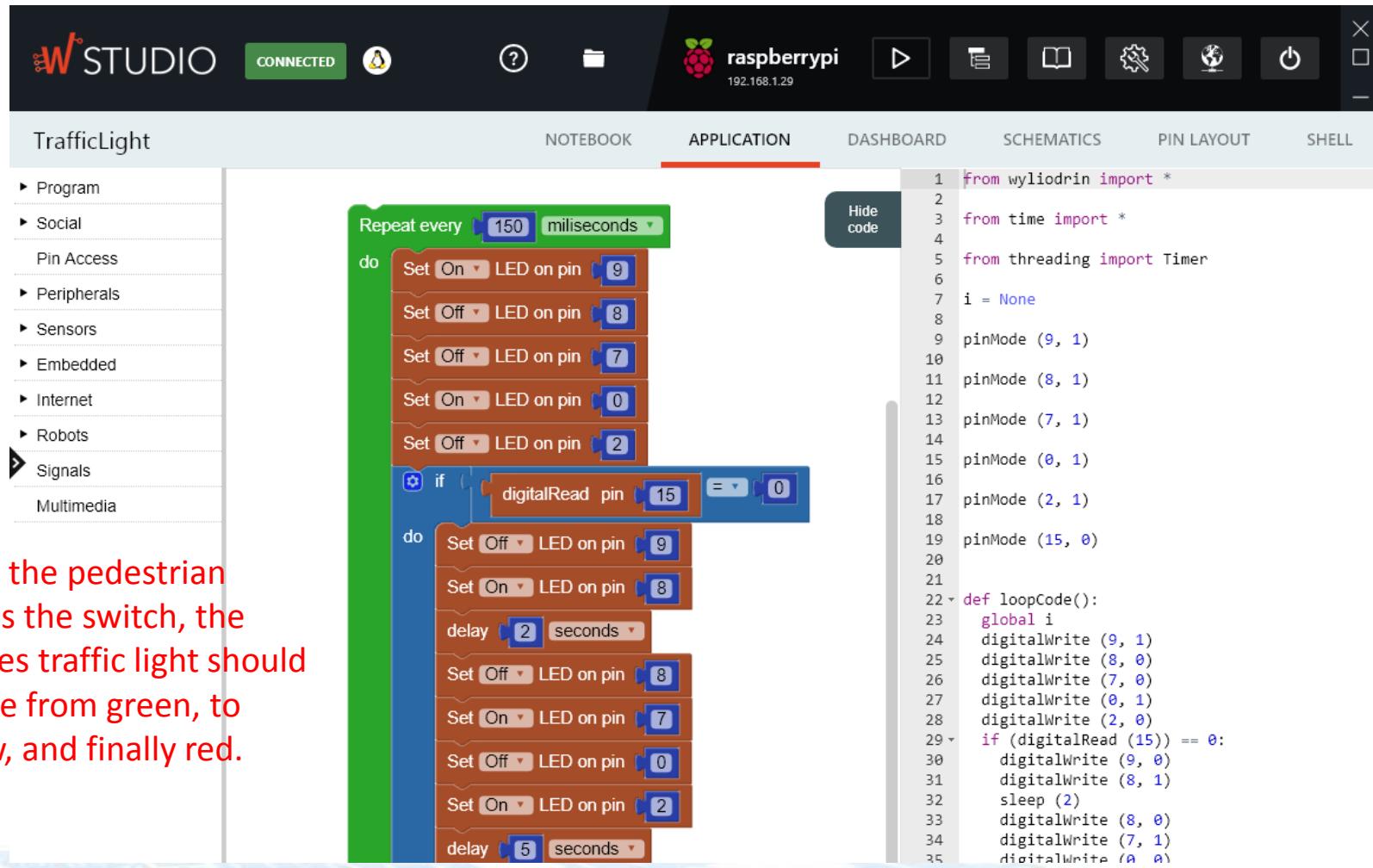
Multimedia allows audio management, streaming, and computer-based speech in your board.



Implementation of Traffic Light Simulator

After this review of coding with blocks,
let's continue with the implementation of
our traffic light simulator...

Implementation (1)



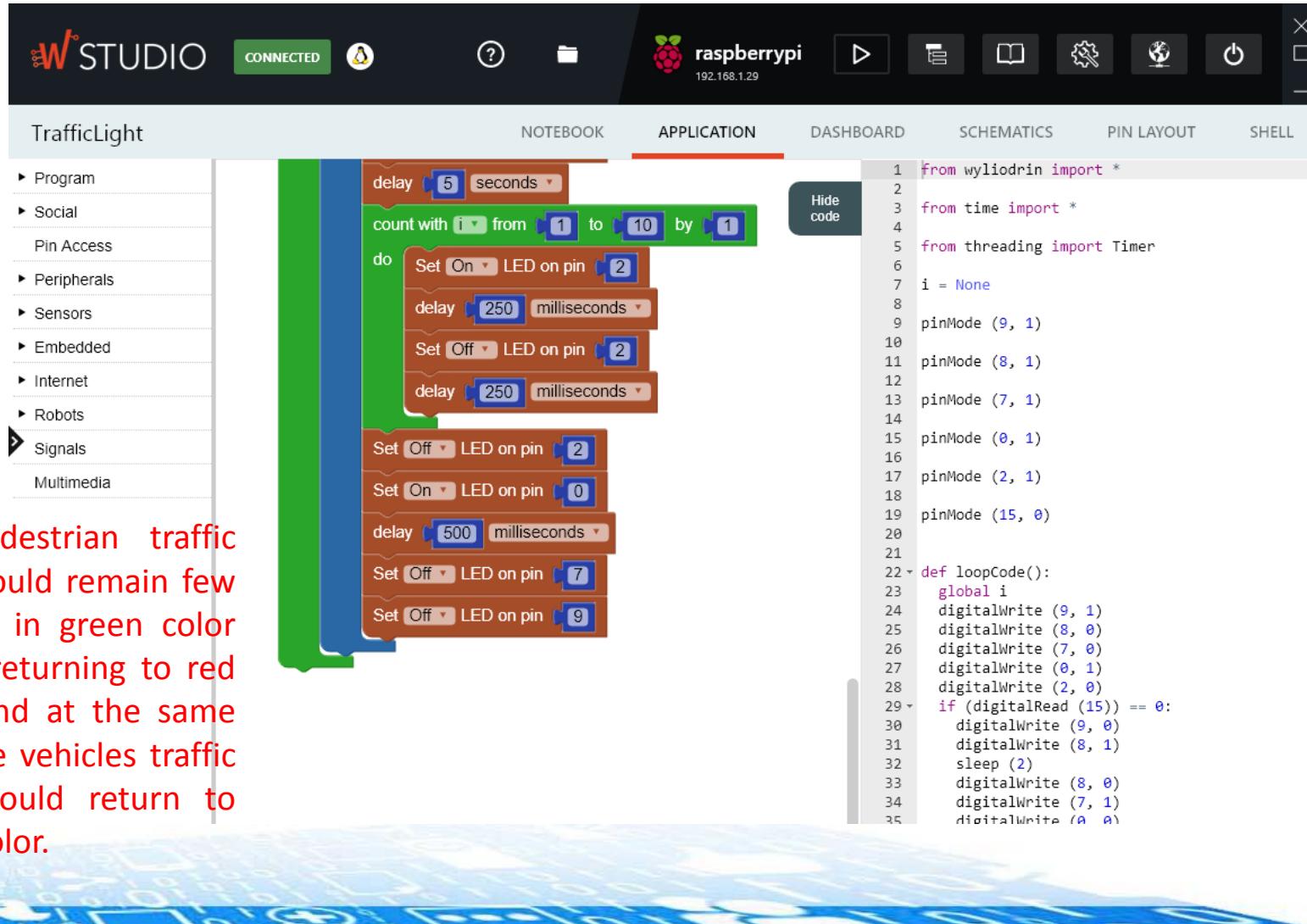
The screenshot shows the WSTUDIO application interface for a Raspberry Pi. The top bar indicates the application is connected to a raspberry pi at 192.168.1.29. The main window displays a Scratch-style script titled "TrafficLight". The script uses a "Repeat every 150 milliseconds" loop to alternate LED states on pins 9, 8, 7, 0, and 2. It includes an "if" condition that changes the sequence based on the state of digital pin 15. The right side of the screen shows the generated Python code:

```
from wylodrin import *
from time import *
from threading import Timer
i = None
pinMode (9, 1)
pinMode (8, 1)
pinMode (7, 1)
pinMode (0, 1)
pinMode (2, 1)
pinMode (15, 0)

def loopCode():
    global i
    digitalWrite (9, 1)
    digitalWrite (8, 0)
    digitalWrite (7, 0)
    digitalWrite (0, 1)
    digitalWrite (2, 0)
    if (digitalRead (15)) == 0:
        digitalWrite (9, 0)
        digitalWrite (8, 1)
        sleep (2)
        digitalWrite (8, 0)
        digitalWrite (7, 1)
        digitalWrite (0, 0)
```

When the pedestrian pushes the switch, the vehicles traffic light should change from green, to yellow, and finally red.

Implementation (2)



The pedestrian traffic light should remain few seconds in green color before returning to red color, and at the same time the vehicles traffic light should return to green color.

WSTUDIO
CONNECTED raspberry pi 192.168.1.29

TrafficLight

APPLICATION

NOTEBOOK DASHBOARD SCHEMATICS PIN LAYOUT SHELL

Program Social Pin Access Peripherals Sensors Embedded Internet Robots Signals Multimedia

delay [5 seconds] count with [i] from [1] to [10] by [1]

do

- Set [On] LED on pin [2]
- delay [250 milliseconds]
- Set [Off] LED on pin [2]
- delay [250 milliseconds]
- Set [Off] LED on pin [2]
- Set [On] LED on pin [0]
- delay [500 milliseconds]
- Set [Off] LED on pin [7]
- Set [Off] LED on pin [9]

from wylodrin import *
from time import *
from threading import Timer
i = None
pinMode (9, 1)
pinMode (8, 1)
pinMode (7, 1)
pinMode (0, 1)
pinMode (2, 1)
pinMode (15, 0)

def loopCode():
 global i
 digitalWrite (9, 1)
 digitalWrite (8, 0)
 digitalWrite (7, 0)
 digitalWrite (0, 1)
 digitalWrite (2, 0)
 if (digitalRead (15)) == 0:
 digitalWrite (9, 0)
 digitalWrite (8, 1)
 sleep (2)
 digitalWrite (8, 0)
 digitalWrite (7, 1)
 digitalWrite (0, 0)

Implementation (3)

- You can play with timing/delay in order to make your traffic light more realistic.
- And print some messages about the status of the traffic lights in the console.
- After you finish your implementation, please demo to TA.

CHECKPOINT 4!



Apendix: VM & Software Tools

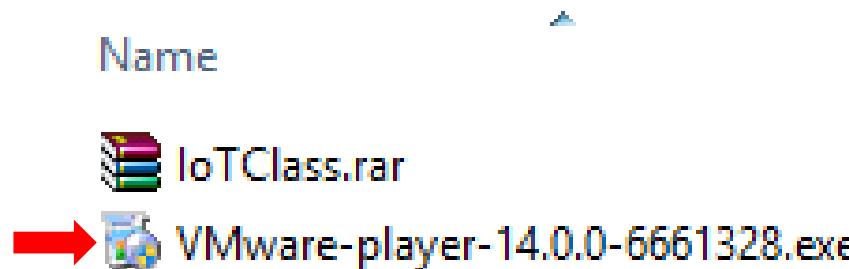
物聯網技術與應用 IoT/M2M Technologies and Applications

國立交通大學資訊工程系
Department of Computer Science
National Chiao Tung University

September 21, 2018

Install VMware

- Copy TA's files to your laptop

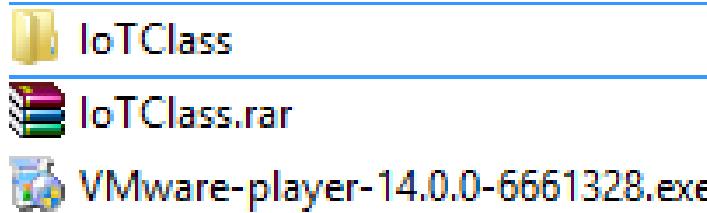
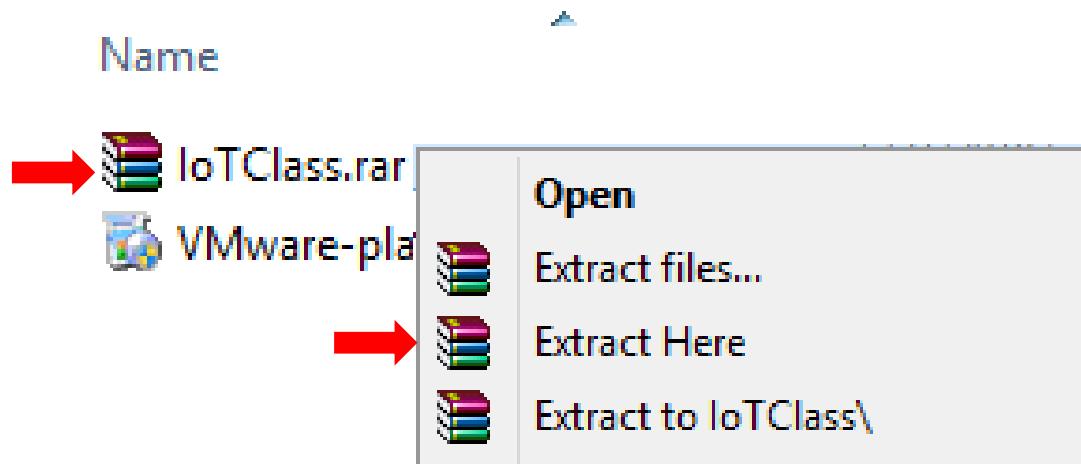


- Install VMware-Player



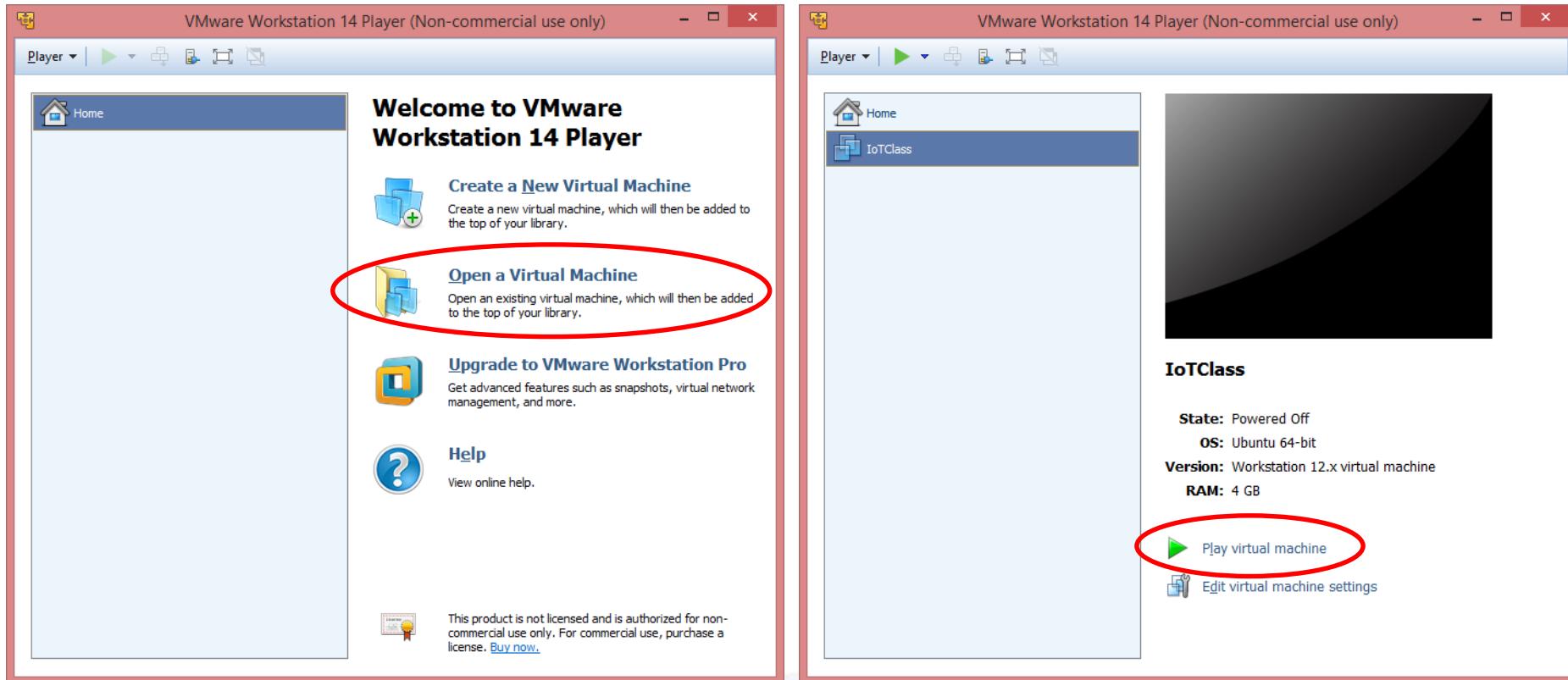
Extract the rar file

- Extract the IoTClass VM.



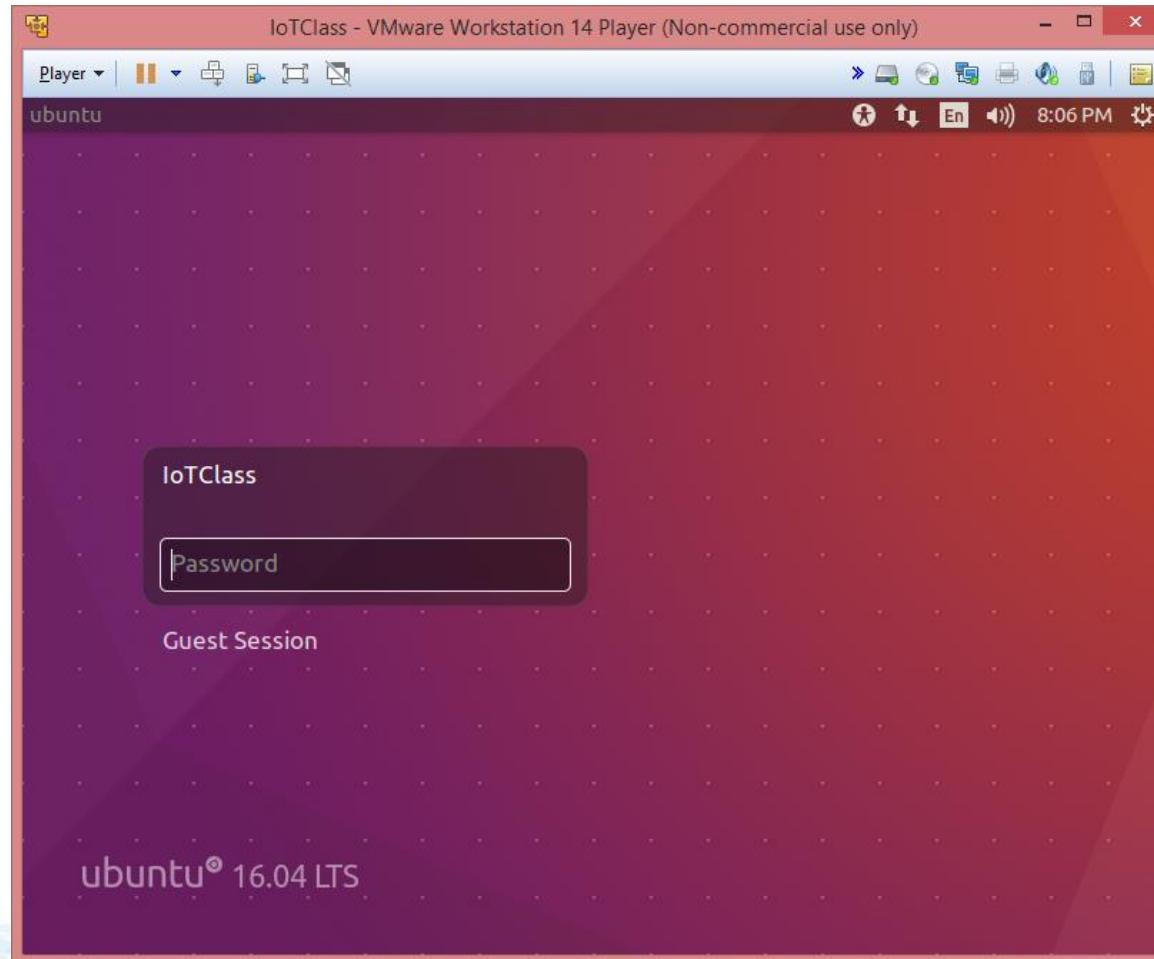
Run the VM

- Run VMware-Player, open the IoTClass VM, and start it.



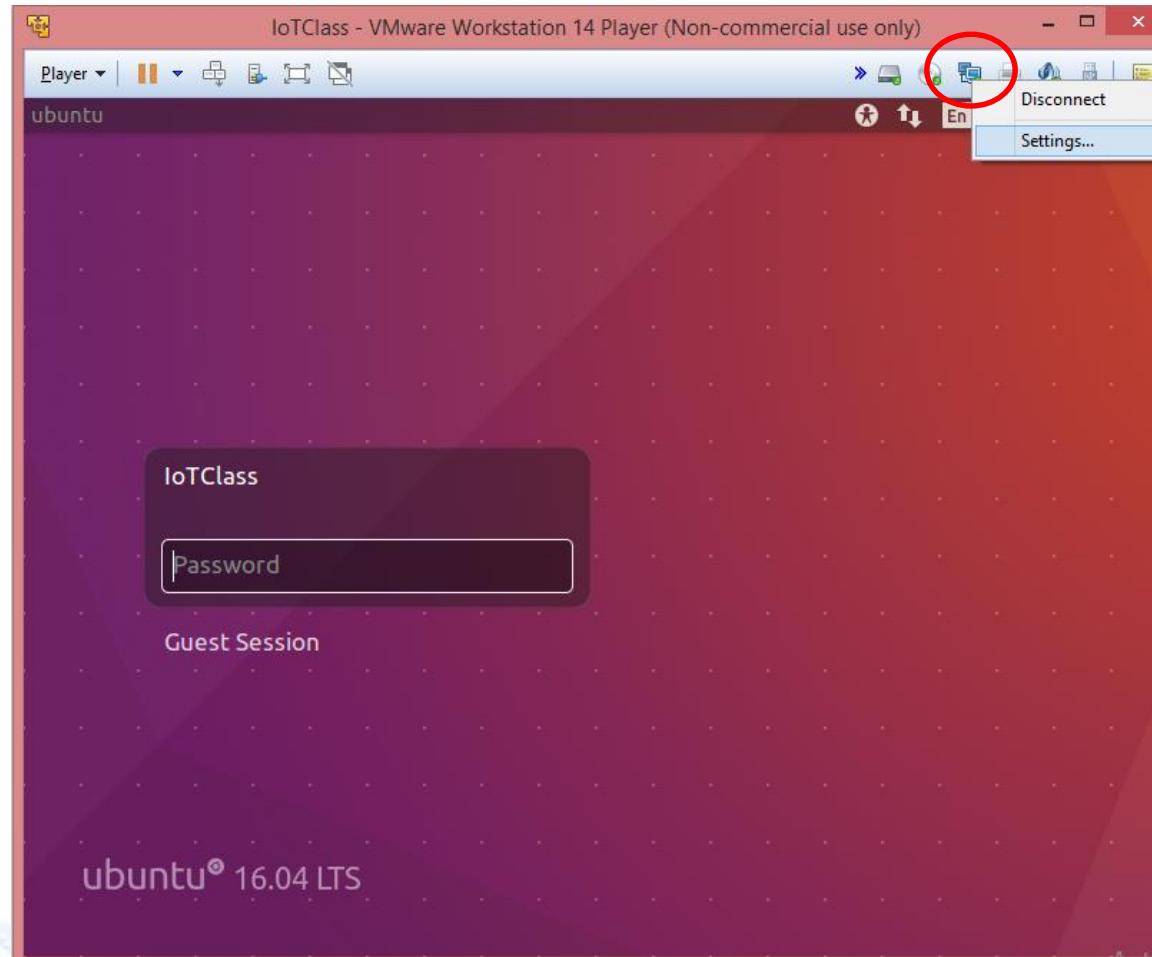
Login into the VM

- Password: **iotclass**



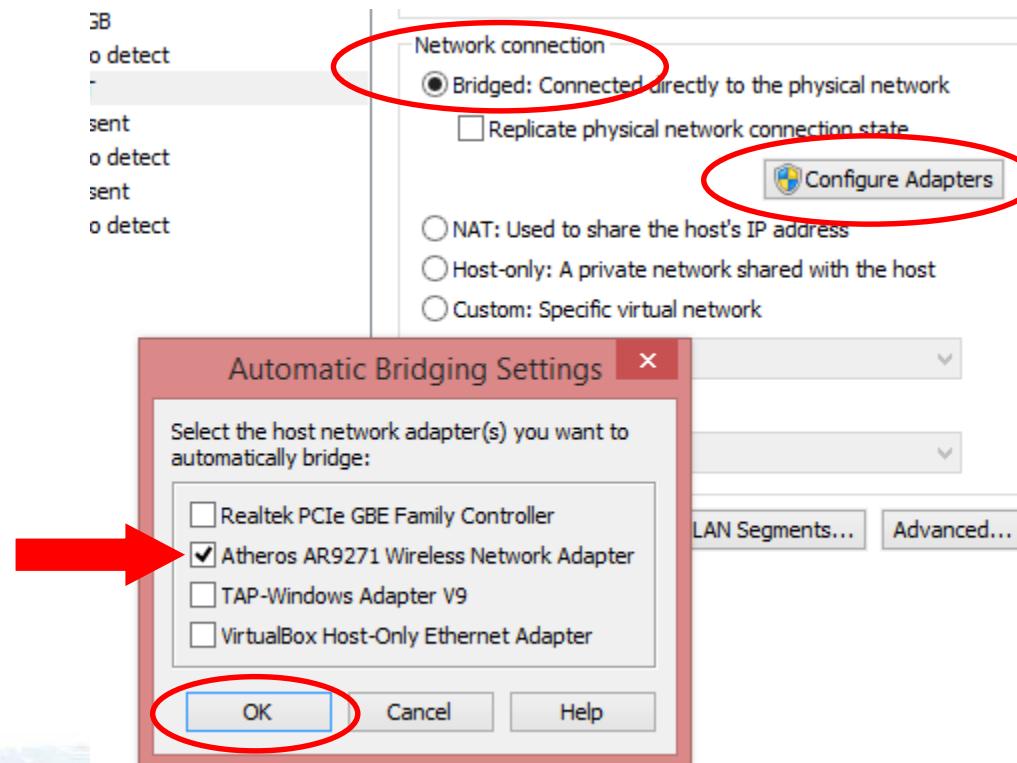
Setup Bridged Network (1)

- Right click on the network icon and choose Settings...



Setup Bridged Network (2)

- Choose **Bridged** radio button.
- Push **Configure Adapters** button.
- Mark your **wireless interface name** checkbox.



Content of the VM

- TA has installed the following software in the VM:
 - Java-JRE
 - Wyliodrin STUDIO
 - OM2M
 - Nodejs
 - node-red
 - contrib-oM2M.gz – a node-red library written by NCTU-LAAS
 - Postman