FUNDAMENTALS OF IOT WITH JAVASCRIPT

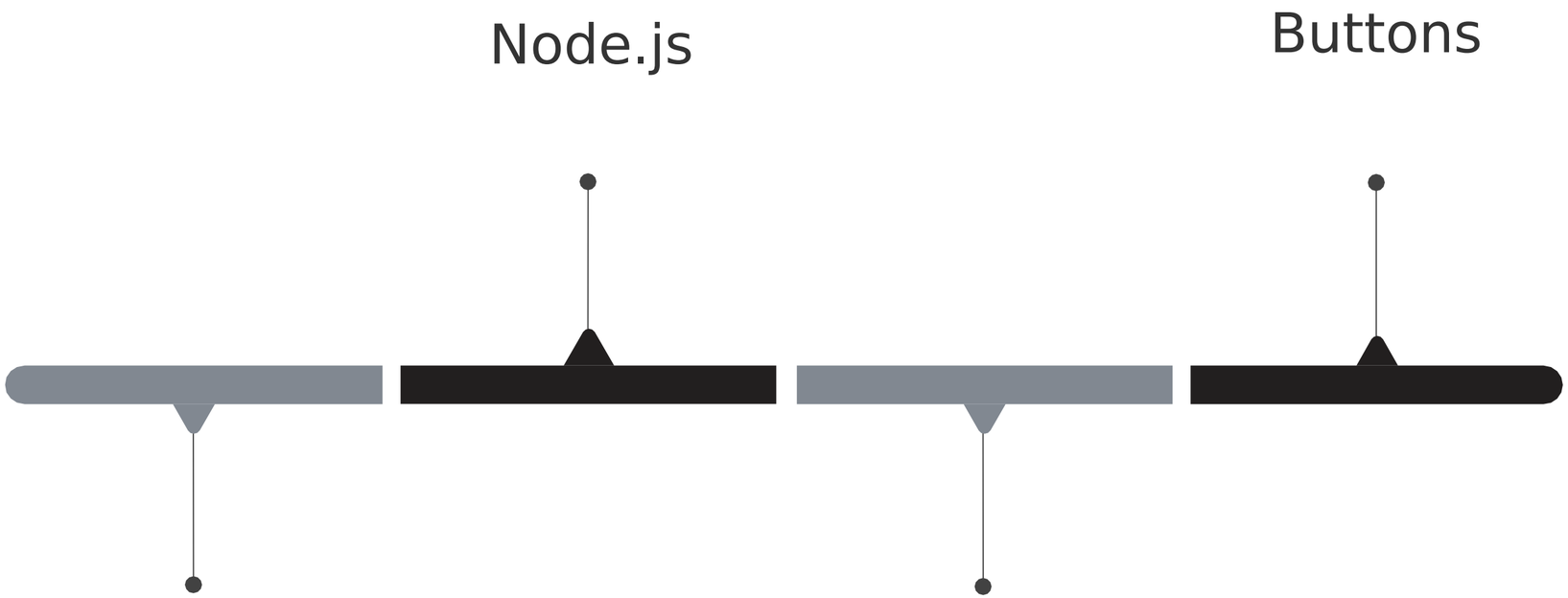
Who am I?

- Senior software engineer and open source enthusiast
- Master of Information Technologies and Bachelor of Computer Systems and Technologies of Technical University Sofia
- 12+ years of professional experience
- Maker of certified open source hardware Internet of Things
- Speaker at various open source events in San Francisco, Portland (OR), Hong Kong, Shanghai, Shenzhen, Brussels, Berlin, Bratislava, Prague, Sofia and Plovdiv

IoT Course

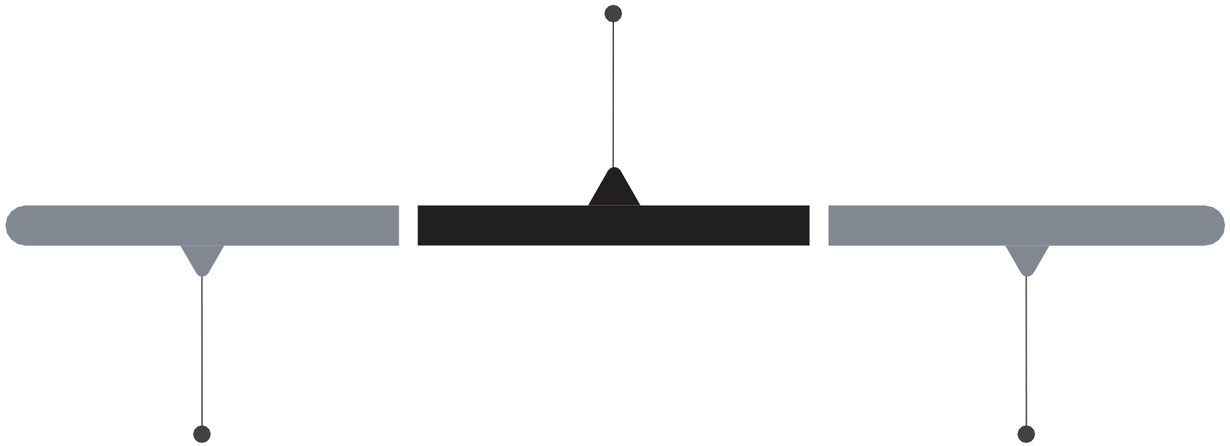
1. 2 Days
2. 7 Sections
3. 7 Lab exercises
4. 150+ slides
5. Examples in GitHub

Roadmap Day 1



Roadmap Day 2

MQTT



Sensor Modules

Integrating it all together!

Section 1: Introducing Raspberry Pi

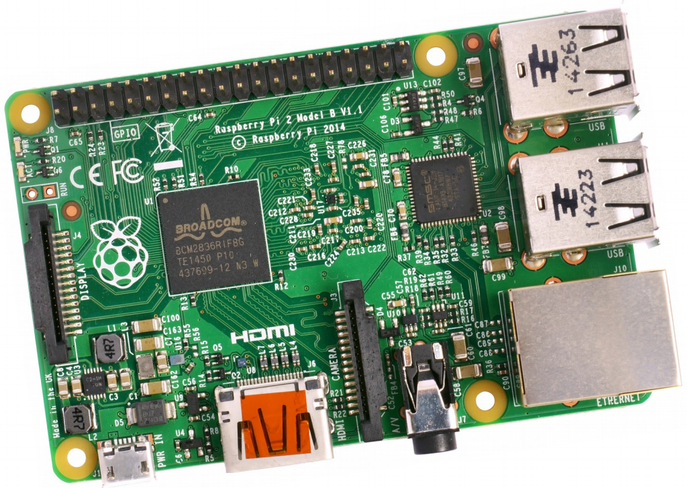
In this lesson you will learn…

- Brief history of Raspberry Pi
- Raspberry Pi versions and models
- Raspbian GNU/Linux distribution
- Peripheral devices and add-on boards for Raspberry Pi

What is Raspberry Pi?

- Low cost single board computer developed in the UK by the Raspberry Pi Foundation
- With Broadcom ARM SoC
- Available with size of credit card (85x56mm), even smaller (65x30mm) or as a industrial compute module
- As of mid 2018 more than 19 million units have been sold worldwide

Raspberry Pi Milestones



- 2009 - Raspberry Pi Foundation
- 2012 - The 1st Raspberry Pi
- 2014 - Raspberry Pi B+ and Raspberry Pi Compute Stick
- 2016 - Raspberry Pi Zero
- 2018 – Raspberry Pi 3 Model B+ with PoE

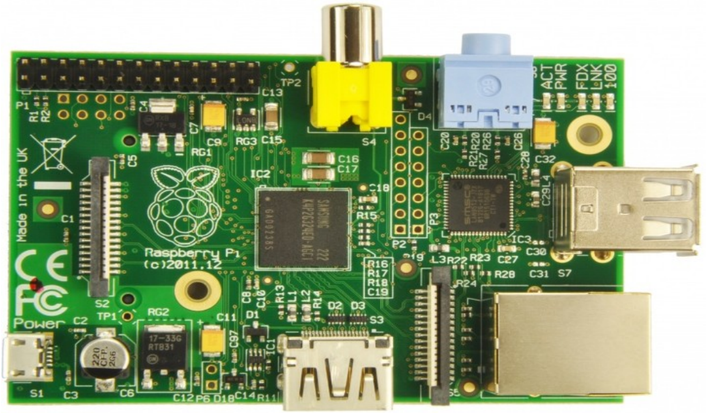
Raspberry Pi Generations

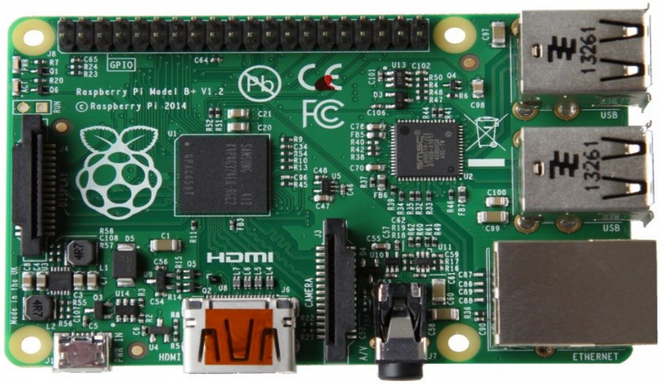
- Raspberry Pi Model A and Model B
- Raspberry Pi Model B+
- Raspberry Pi 2 Model B
- Raspberry Pi 3 Model B
- Raspberry Pi 3 Model B+
- Raspberry Pi 0
- Raspberry Pi 0 W

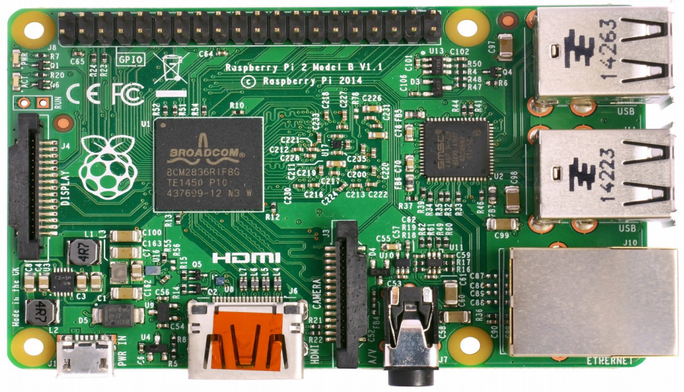
Industrial Compute Modules

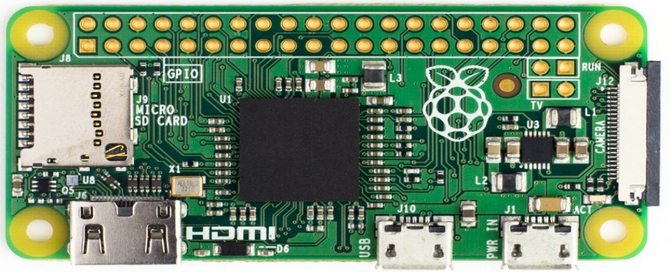
- Raspberry Pi Compute Module 1
- Raspberry Pi Compute Module 3 Lite
- Raspberry Pi Compute Module 3
- Raspberry Pi Compute Module IO Board

Raspberry Pi Flavors



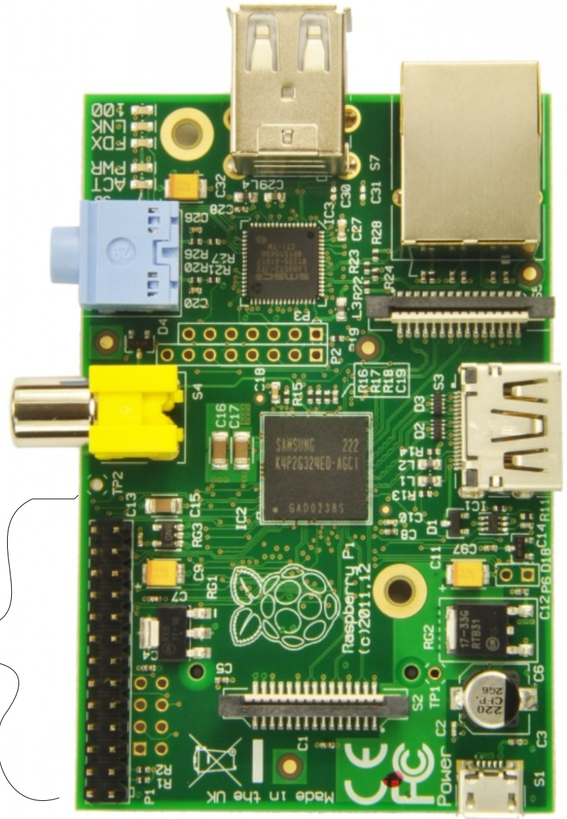






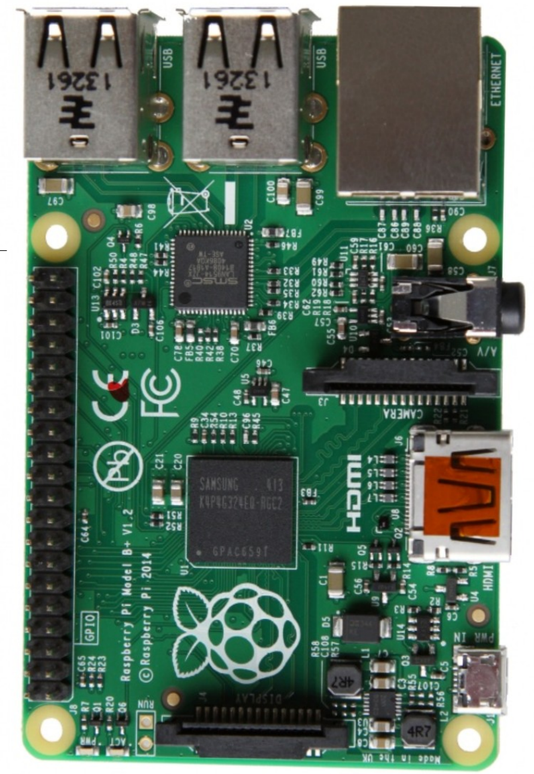
Important change in B+

26 pins



Raspberry Pi Model B (2012)

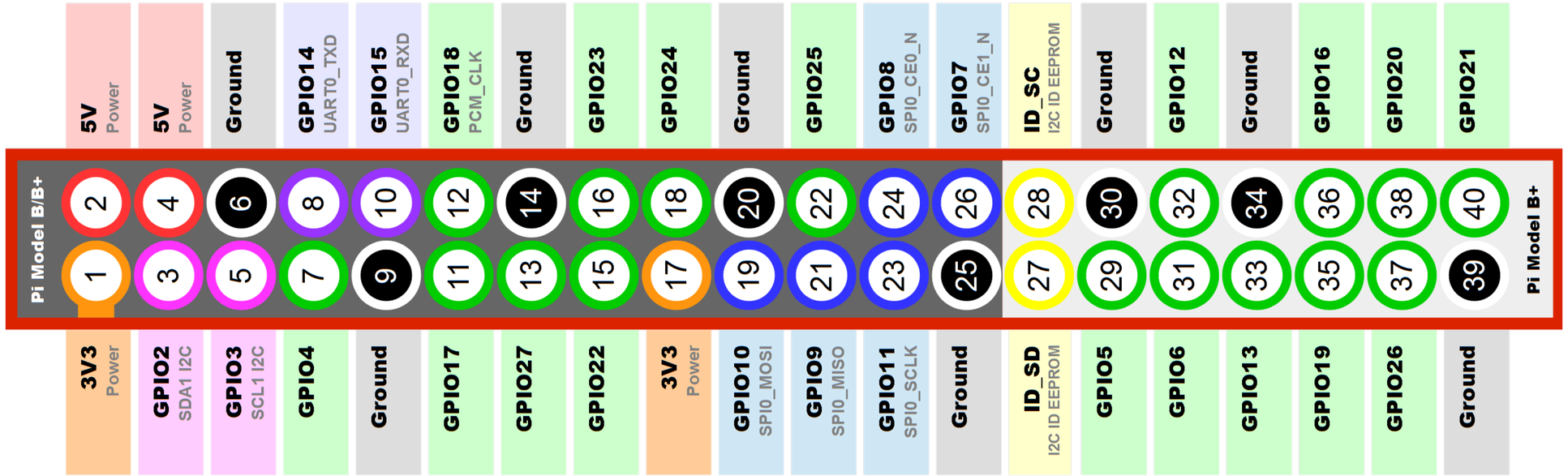
40 pins



Raspberry Pi Model B+ (2014)

Raspberry Pi Pinout

- 40 pin header
- Specific pins for I2C, SPI, PWM and serial

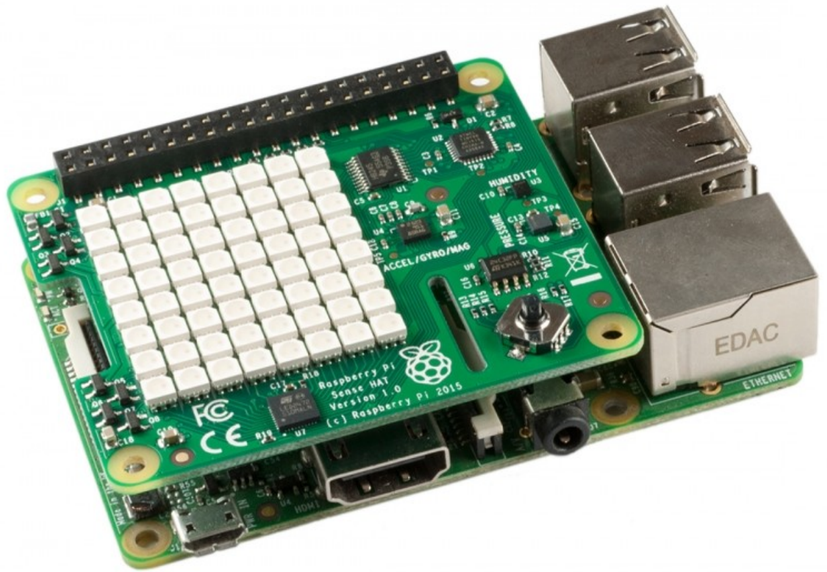


Official Peripherals

- 7” official touchscreen display for Raspberry Pi
- Camera modules
- Raspberry Pi Sense HAT
- Raspberry Pi PoE HAT

Sense HAT

- Official product of the Raspberry Pi Foundation
- Sensors for temperature, humidity, barometric pressure, gyroscope, accelerometer, magnetometer
- 8x8 RGB LED matrix
- Five-button joystick

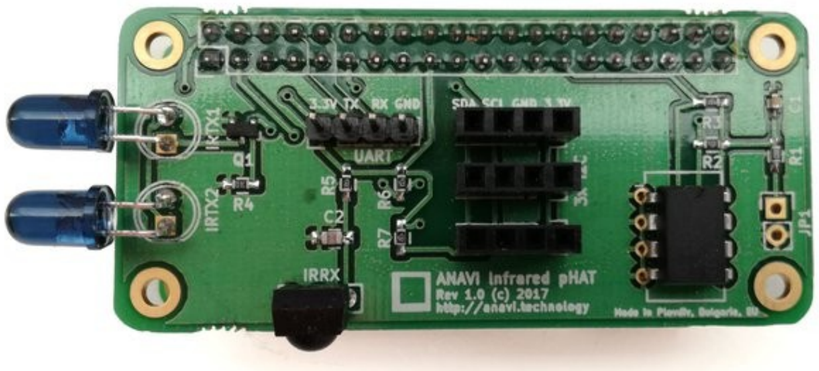


What is HAT?

- **H**ardware **A**ttached on **T**op (HAT)
- Form factor and dimensions (65x56mm)
- 40 pin header compatible with Raspberry Pi B+ and the newer versions
- EEPROM with device tree fragment connected to the secondary I2C bus on pins 27 and 28
- Details:
  https://www.raspberrypi.org/blog/introducing-raspberry-pi-hats/
  https://github.com/raspberrypi/hats

What is pHAT?

- Form factor suitable for Raspberry Pi Zero with 4 mount holes and dimensions 65x30mm
- 40 pin through-hole header
- EEPROM **not** mandatory

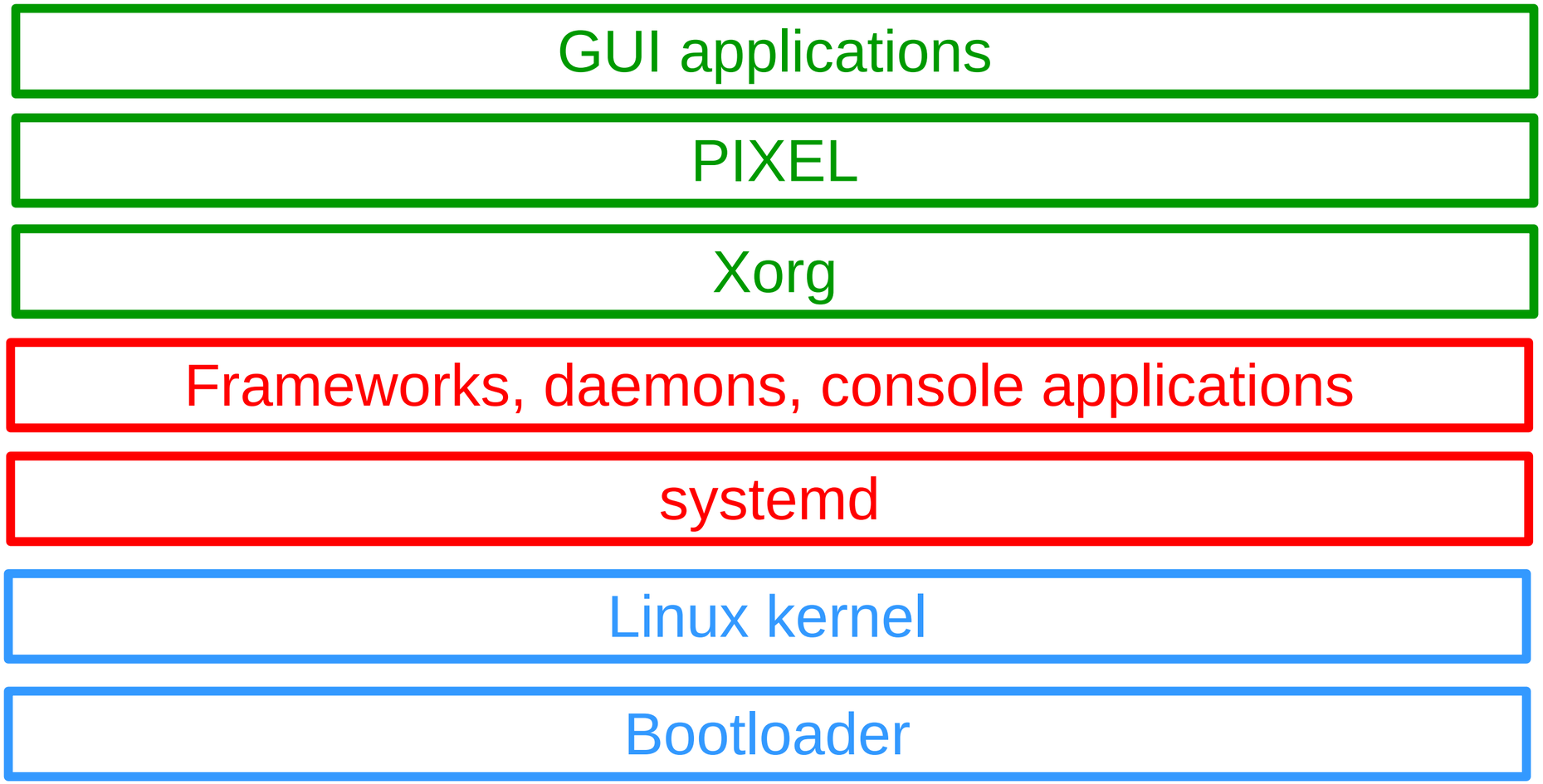


- Not an official standard of the Raspberry Pi Foundation

Raspbian

- Free and open source GNU/Linux distribution based on Debian
- Optimized for the Raspberry Pi hardware
- Over 35,000 packages and pre-compiled software
- Available for free download at: https://www.raspberrypi.org/downloads/

Raspbian


GUI applications
PIXEL
Xorg
Frameworks, daemons, console applications
systemd
Linux kernel
Bootloader

Raspbian Configurations

- **config.txt** – configuration text file, acting instead of a BIOS of a personal computer, stored on the boot partition of the Raspberry Pi microSD card with format **property=value**
- **raspi-config** – command line configuration tool with text-based user interface, allowing you to easily enable features such as I2C, SPI, UART, SSH, VNC, camera and many more

In this lesson you learned...

- What is Raspberry Pi
- What is HAT and pHAT
- What is Raspbian

Questions?

Lab 1:

Installing Raspbian on Raspberry Pi 3

Scenario:

Download and flash Raspbian on microSD card
Turn on Raspberry Pi and finalize the installation

Aim:

Prepare a working environment on Raspberry Pi

Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card
- Personal computer (with Windows, Mac OS or a Linux distribution)

Steps

1) Download Etcher https://etcher.io/
2) Download Raspbian https://www.raspberrypi.org/downloads/raspbian/
3) Flash Raspbian on microSD card using Etcher
4) Attach keyboard, mouse, HDMI monitor to your Raspberry Pi
5) Plug the microSD card in Raspberry Pi and turn it on using 5V USB power supply
6) Connect to a WiFi network
7) Open a terminal, run **sudo raspi-config** and enable SSH and I2C

Section 2: Introducing Node.js

In this lesson you will learn...

What is Node.js?

What is NPM and how to use it?

What is JavaScript?

- Interpreted programming language
- Supports event-driven, functional, and object-oriented (prototype-based) programming styles
- First released in 1995 as a scripting language for Netscape
- Designed by Brendan Eich

JavaScript Standardization

- Started in 1996 by standards organization Ecma International
- ECMAScript (or ES) is a trademarked name of a scripting-language specification standardized by Ecma International in ECMA-262 and ISO/IEC 16262
- ECMA-262 has 9 editions, including ECMAScript 2015 (6th edition), ECMAScript 2016 (7th edition), ECMAScript 2017 (8th edition) and ECMAScript 2018 (9th edition)

JavaScript Popularity

- Extremely popular for front-end development
- Continuous increasing popularity with highest position in the TIOBE programming community index during January 2018
- Language of the Year 2014 according to the TIOBE programming community index

What is Node.js?

- Run-time environment to execute JavaScript code outside of a browser
- Cross-platform, works on Windows, Mac OS, FreeBSD and Linux
- **Free and open source**, source code available at GitHub: https://github.com/nodejs/
- Written in C, C++ and JavaScript
- Initially released on 27 May 2009

Node.js Community

- Node.js is a trademark of Joyent, Inc
- Governed by the Node.js Foundation
- Facilitated by the Linux Foundation's Collaborative Projects program
- Source code in GitHub with more than 2 thousand contributors to the core components
- Thousands of free and open source packages

What are Node.js main features?

- Built on Chrome's V8 JavaScript engine.
- Uses asynchronous programming
- Has built-in modules (such as fs, http, https, os, etc.)
- Uses a package manager for installing dependencies
- Learn more at: https://nodejs.org/en/docs/

What is NPM?

- Package manager for Node.js packages
- npmjs.com hosts thousands of free and open source Node.js packages to download and use
- npmjs offers paid subscription services for hosting private packages and integration of npm for enterprises
- Getting started guidelines: https://docs.npmjs.com/getting-started/

How to use NPM?

- Install npm package:
  npm install upper-case

- Load and use it in your JavaScript source code:
  var myNewPackage = require('upper-case');

How to create NPM Package?

- Node.js package includes package.json, README.md and source code
- Use **npm init** to create package.json
- Create text file README.md with details about the new package
- Optionally, publish the package to npmjs.com: https://docs.npmjs.com/getting-started/publishing-npm-packages

How to install Node.js on Raspberry Pi?

- Update the system package list:
  sudo apt-get update
- Install Node.js:
  sudo apt-get install -y nodejs
- Verify the installation:
  node -v
- Example output:
  pi@raspberrypi:~ $ node -v
  v8.11.1

One more thing… open source software

- Source code is released under a license in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose
- Various different open source licenses define how exactly the software should be used and what are the terms under which it can be changed
- Popular open source licenses: GPL, LGPL, MIT, Apache license, BSD license, Eclipse Public License, Mozilla Public License, etc.

In this lesson you learned...

- The brief history of JavaScript
- What is Node.js
- What is NPM
- How to install Node.js on Raspberry Pi

Questions?

Lab 2: Getting started with Node.js on Raspberry Pi

Scenario:

Install Node.js on Raspberry Pi (with Raspbian)
Write hello.js and Node.js package to install it.

Aim:

Create a simple “Hello, World” Node.js package

Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card with Raspbian

Steps

1) Open a terminal and type in the following commands to install Node.js on Raspberry Pi:
sudo apt-get update
sudo apt-get install -y nodejs npm git

2) Create directory **lab2-hello** and file **hello.js** with the following content:
#! /usr/bin/env node
console.log('Hello, world!');

3) Create **README.md** and add brief text description of the project

4) Run **npm init** and follow the on-screen instructions to create **package.json**

5) Add the following line to package.json to add the executable to the global PATH of the system:
"bin": "./hello.js",

6) Install the application:
sudo npm install -g

package.json

```
{
 "name": "lab2-hello",
 "version": "1.0.0",
 "description": "Hello, World",
 "main": "hello.js",
 "bin": "./hello.js",
 "scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
 },
 "repository": {
  "type": "git",
  "url": "git+https://github.com/leon-anavi/iot-javascript-rpi.git"
 },
 "keywords": [
  "hello"
 ],
 "author": "Leon Anavi",
 "license": "MIT",
 "bugs": {
  "url": "https://github.com/leon-anavi/iot-javascript-rpi/issues"
 },
 "homepage": "https://github.com/leon-anavi/iot-javascript-rpi#readme"
}
```

hello.js

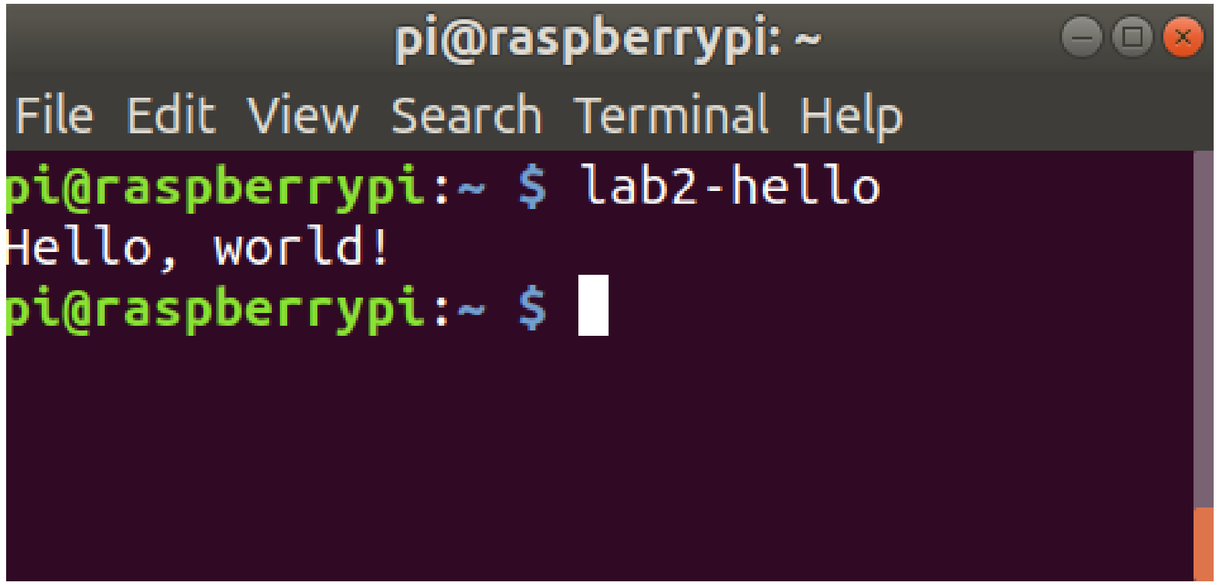
```
#! /usr/bin/env node

console.log('Hello, world!');
```

Demo

- Source code available at:
  https://github.com/leon-anavi/iot-javascript-rpi/tree/master/lab2-hello

- Demo:
  pi@raspberrypi:~ $ lab2-hello
  Hello, world!



BREAK

10min

Section 3: LED

In this lesson you will learn...

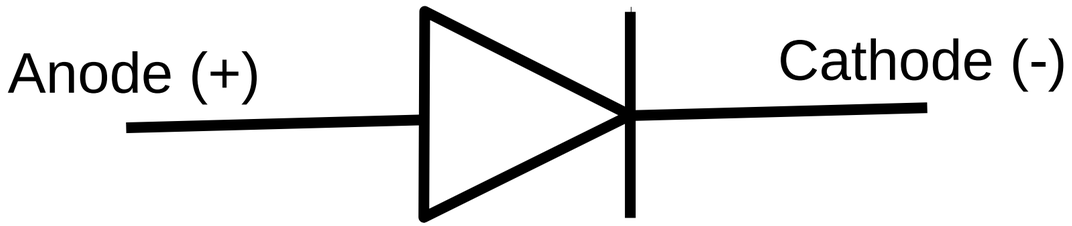
Understanding how LED works

Wiring a LED to Raspberry Pi

Using Node.js for making a blinking LED

What is a diode?

- Electronic component to control the direction of the current flow
- Diode conducts current only in one direction (forward direction) and blocks the current trying to flow in the opposite direction
- The symbol used to represent a diode in circuit diagram is a triangle and a line, the triangle points in the direction of the current

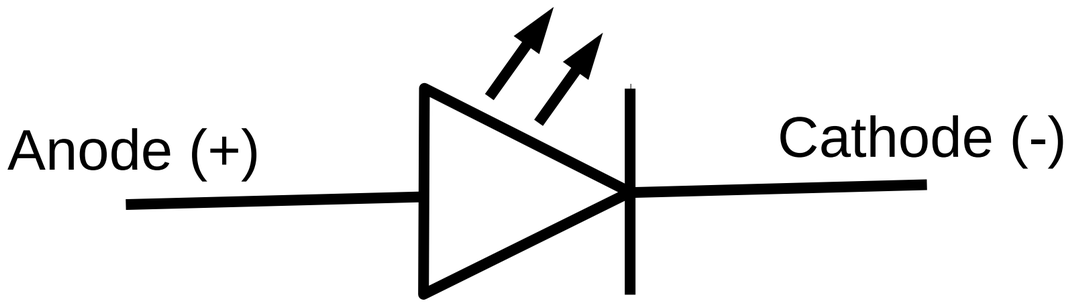


What is a LED?

- Light-Emitting Diode (LED) is a particular type of diode that converts electrical energy into light
- LEDs are energy efficient and don’t get hot as conventional lightbulbs
- LEDs comes in huge variety of colors, sizes and shapes
- LEDs are very popular electronic components used in almost any embedded device.

What is a the symbol of a LED?

- The symbol used to represent a LED in circuit diagram looks like a diode (triangle and a line) with a couple of arrow going out of the triangle
- The triangle points in the direction of the current

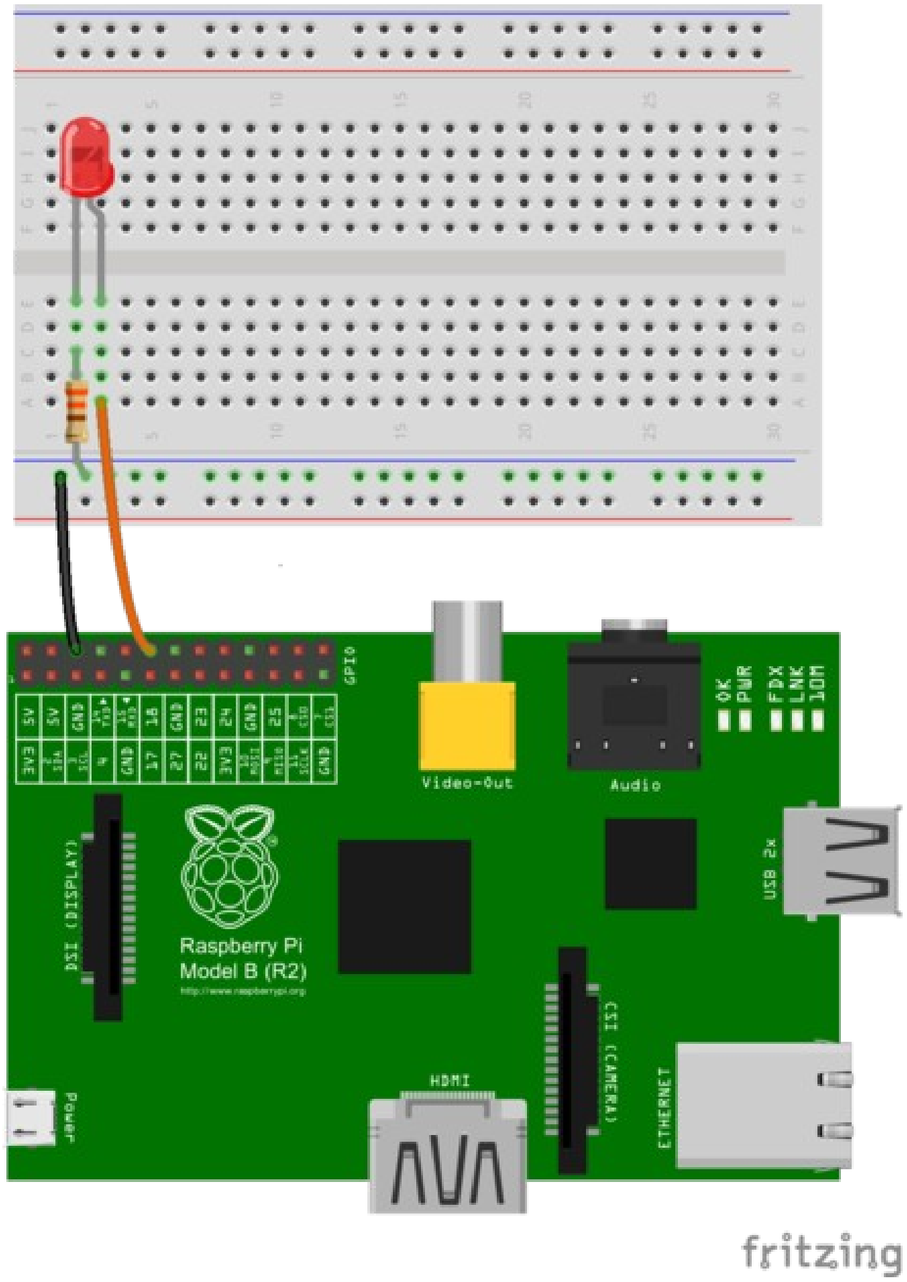


Diffused vs clear LED

- Diffused LED comes in a housing made from an epoxy resin, the housing looks cloudy and spreads the light in a wide viewing range
- Clear LED has a higher brightness and lower viewing angle compared to diffused LED

How to connect a LED to Raspberry Pi?

- Connect the shorter leg (cathode) to ground (GND)
- Connect the longer leg to a resistor and to a GPIO pin of Raspberry Pi
- Pick a resistor depending on the datasheet of the LED, recommended is 330 Ohm (Ω)



How to switch on and off an LED?

- Use the Node.js module **onoff** to control GPIO on Raspberry Pi
  var gpio = require('onoff').Gpio;
- Set the GPIO of the Raspberry Pi to output
  var red = new gpio(13, 'out');
- Let current flow from the GPIO to turn **ON** the LED
  red.writeSync(1);
- Block the current flow from the GPIO to turn OFF the LED
  red.writeSync(0);

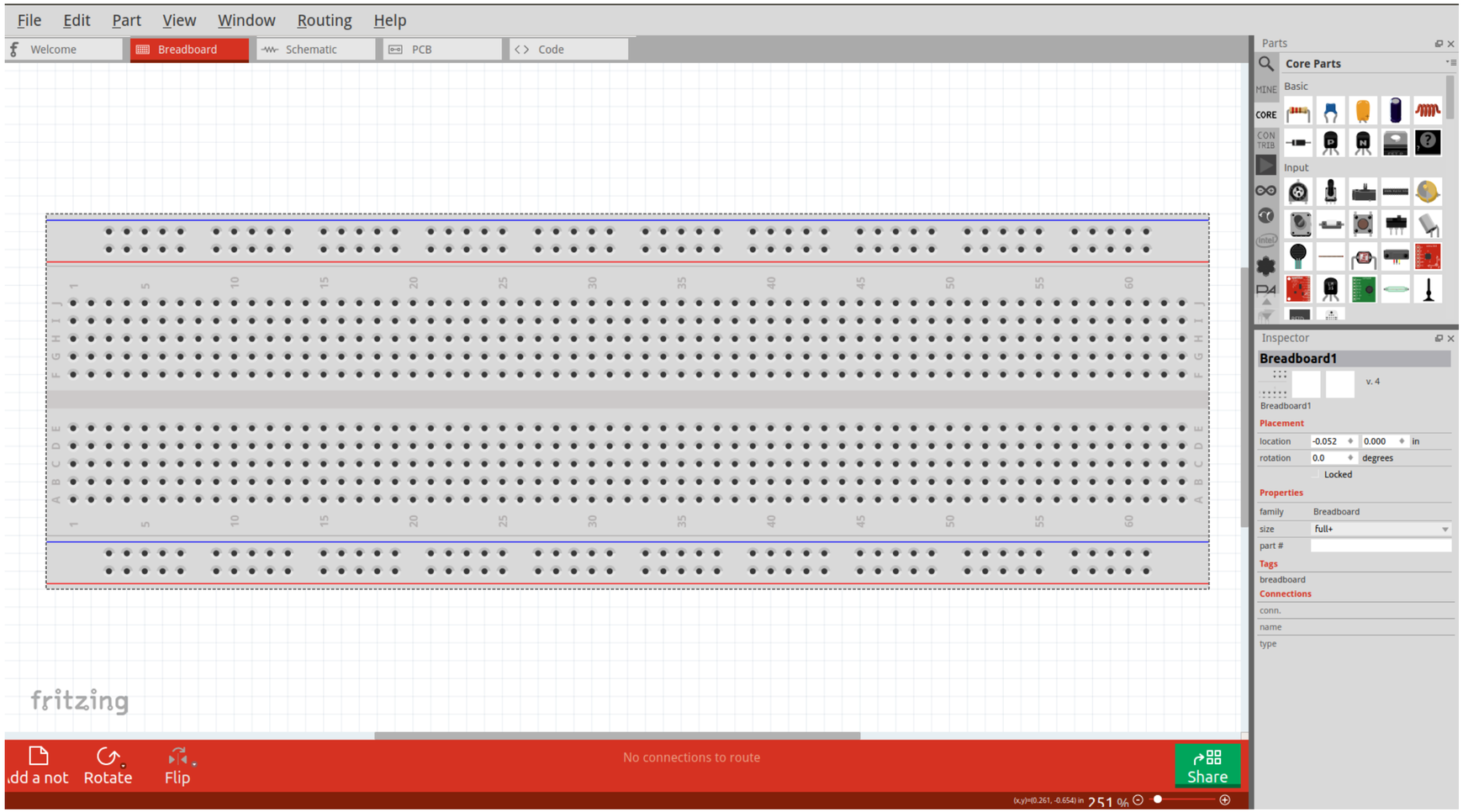
One more thing... resistor

- Electrical component that implements electrical resistance as a circuit element in an electronic circuit
- The SI unit of electrical resistance is the ohm (Ω)
- The symbol used to represent a diode in circuit diagram looks like a rectangle

Fritzing

- Free and open source software for simple and quick design of electronics hardware and wiring diagrams
- Source code written in C++ using the Qt framework
- Available at GitHub under GPLv3
- Available for MS Windows, Mac OS X and Linux distributions
- Download at: http://fritzing.org/download/

Fritzing


File Edit Part View Window Routing Help
Welcome
Breadboard
Schematic
PCB
Code
Parts
Core Parts
Basic
Input
Inspector
Breadboard1
v. 4
Breadboard1
Placement
location -0.052 0.000 in
rotation 0.0 degrees
Locked
Properties
family Breadboard
size full+
part #
Tags
breadboard
Connections
conn.
name
type
fritzing
Add a not
Rotate
Flip
No connections to route
Share

Slide10

In this lesson you learned...

- What is LED and how does it work
- How to connect a LED to Raspberry Pi
- How to control a LED on Raspberry Pi with Node.js

Questions?

Lab 3: Making a traffic light with Node.js on Raspberry Pi

Scenario:

Attach red, green and yellow LED to Raspberry Pi 3
Write Node.js applications that turns on and off each LED periodically

Aim:

Learn how to control LED with Node.js

Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card with Raspbian
- ANAVI Traffic Lights (or a breadboard with jumper wires, 3 resistors and red, yellow and green 5mm LED)

Steps

1) Create directory **lab3-traffic-light** and file **traffic-light.js** with the following content as in https://github.com/leon-anavi/iot-javascript-rpi/blob/master/lab3-traffic-light/traffic-light.js
2) Create **README.md** and add brief text description of the project
3) Run **npm init** and follow the on-screen instructions to create **package.json**
4) Add the following line to package.json to add the executable to the global PATH of the system:
"bin": "./traffic-light.js",
5) Add the following line to list dependencies in package.json:

```
"dependencies": {
   "onoff": ">=3.2.1"
},
```

6) Install the application:
sudo npm install -g

Demo

- Run the application:
  lab3-traffic-light
- Verify that each of the green, yellow and red LEDs is turned on for a second
- Press Ctrl+C to terminate the application and verify that after that all LEDs are turned off

Section 4: Buttons

In this lesson you will learn...

Understanding how a button works

Wiring a button to Raspberry Pi

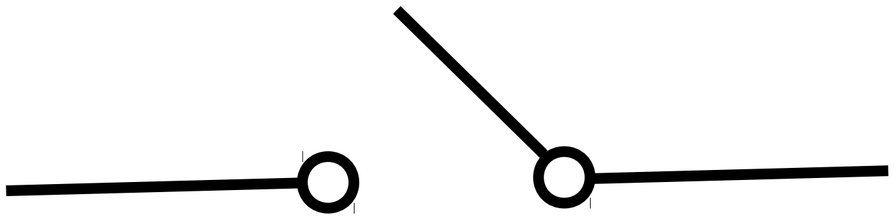
Detecting when a button has been pressed with Node.js

What is a switch?

- Electronic component that controls the current flow in a electrical circuit
- Has two states: “open” (the switch is nonconducting) or “closed” (the switch is conducting and the current is flowing)
- Can exists only in one state
- There are different types of switches: push buttons, toggle, rotary, rocket, etc.

What is the symbol of a switch?

- The symbol used to represent a switch in circuit diagram consists of three lines, two of which are connected together with obtuse angle
- The simplest on-off switch has two terminals which are either disconnected if the switch is “open” or connected if the switch is “closed”

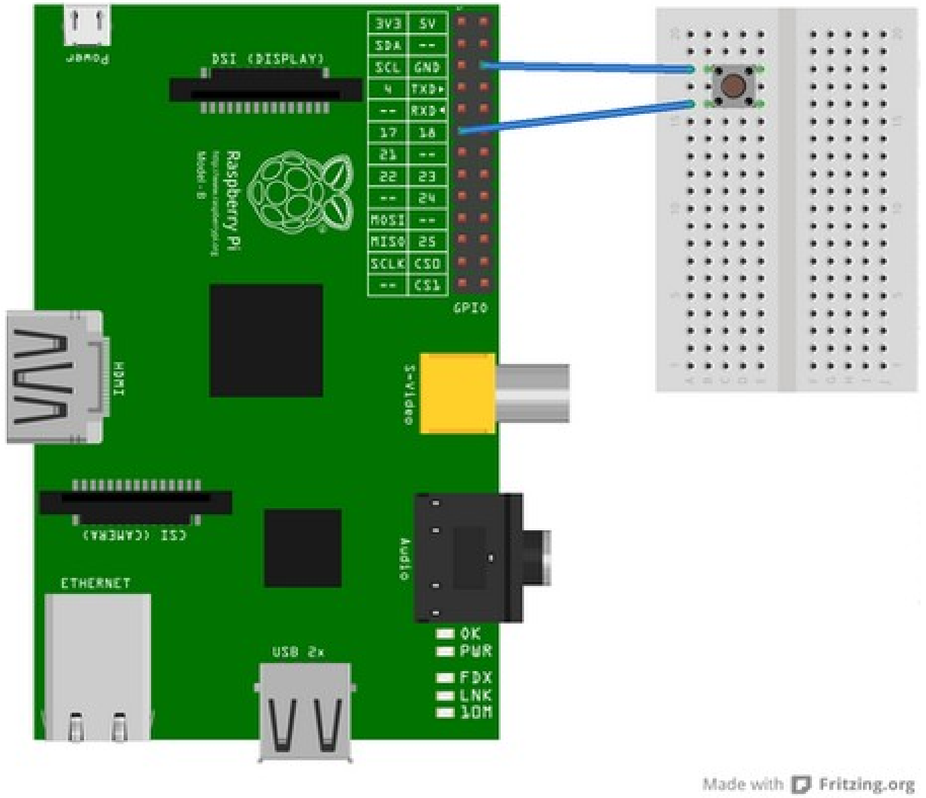


Momentary vs maintained

- Momentary switch which remains active only while they are actuated, for example while the user is pressing a push button
- Maintained switch, also called toggle, keeps its state until actuated into a new one, for example a conventional on-off wall switch for lights

How to connect a tact button to Raspberry Pi?

- The tack push button is a momentary switch
- Connect the one end of the switch to ground (GND) and the other to a GPIO on the Raspberry Pi



How to detect when a button has been pressed with Node.js? (1/2)

- Use the Node.js module **onoff** to read data from a GPIO on Raspberry Pi
  var gpio = require('onoff').Gpio;

- Set the GPIO of the Raspberry Pi to input state
  var button = new gpio(10, 'in', 'both', {debounceTimeout: 500});

- Set debouncing timeout to avoid the issues when the hardware is thinking that a button was pressed several times although it was only pressed once

How to detect when a button has been pressed with Node.js? (2/2)

- Watch for hardware interrupts when the button has been physically pressed:

```
button.watch(function (err, value) {
if (err) {
      console.error('Error: ', err);
      return;
   }
   if (0 == value) {
      console.log("Button pressed");
   }
});
```

One more thing… open source hardware

- Design of physical objects that is publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design
- The Open Source Hardware Association (OSHWA) maintains the Open Source Hardware certification https://www.oshwa.org/

Slide10

In this lesson you learned...

- How buttons works
- How to connect a button to Raspberry Pi
- How to detect when a button has been pressed on Raspberry Pi with Node.js

Questions?

Lab 4: Controlling a button with Node.js on Raspberry Pi

Scenario:

Attach a button and a LED to Raspberry Pi 3
Write Node.js applications that turns on and off the LED when the button is pressed using npm package onoff

Aim:

Understand how to interact with a button

Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card with Raspbian
- ANAVI Buttons (or a breadboard with jumper wires and a tact button)
- ANAVI Traffic Lights (or a breadboard with jumper wires, 3 resistors, red, yellow and green 5mm LED)

Steps

1) Create directory **lab4-button** and file **button.js** with the following content as in https://github.com/leon-anavi/iot-javascript-rpi/blob/master/lab4-button/button.js

2) Create **README.md** and add brief text description of the project

3) Run **npm init** and follow the on-screen instructions to create **package.json**

4) Add the following line to package.json to add the executable to the global PATH of the system:
"bin": "./button.js",

5) Add the following line to list dependencies in package.json:

```
"dependencies": {
   "onoff": ">=3.2.1"
},
```

6)Install the application:
sudo npm install -g

Demo

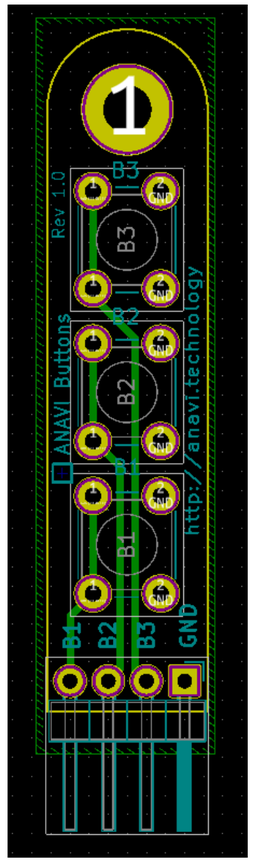
- Run the application:
  lab4-button
- Press button B1 on ANAVI Buttons (or on the breadboard)
- Verify that the output in the terminal shows *Button pressed*
- Verify that the LED turns on and off when the button is pressed
- Press Ctrl+C to terminate the application

Homework

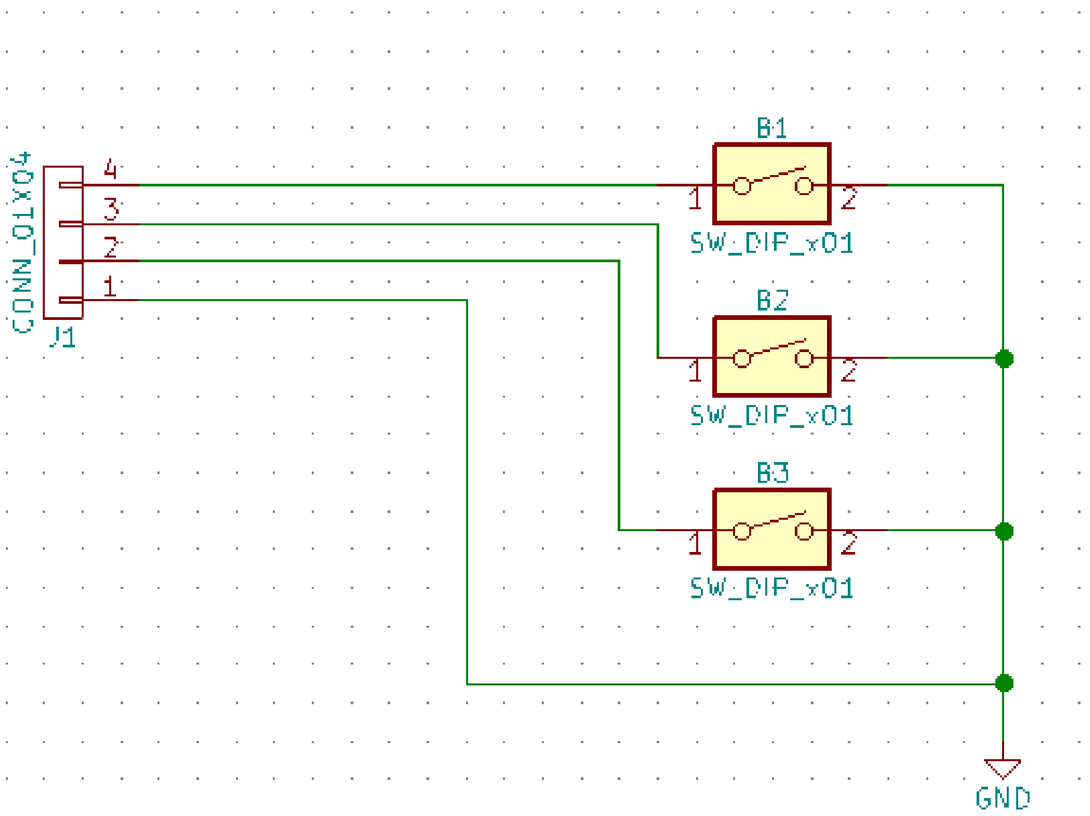
Extend the source code of the application to turn on and off the yellow LED by button B2 and the red LED by button B3.

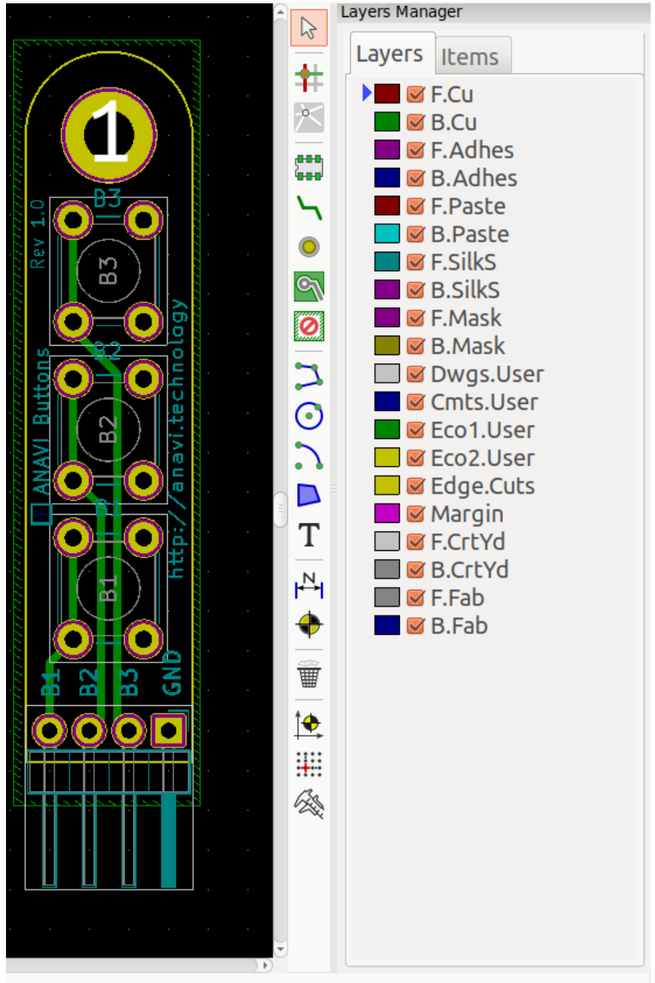
Advanced Homework

- Download and install the free and open source software KiCad: http://kicad-pcb.org/
- Clone the certified open source hardware project ANAVI Buttons from GitHub: https://github.com/AnaviTechnology/anavi-buttons
- Open ANAVI Buttons in KiCad and explore the schematics and the layout of the printed circuit board

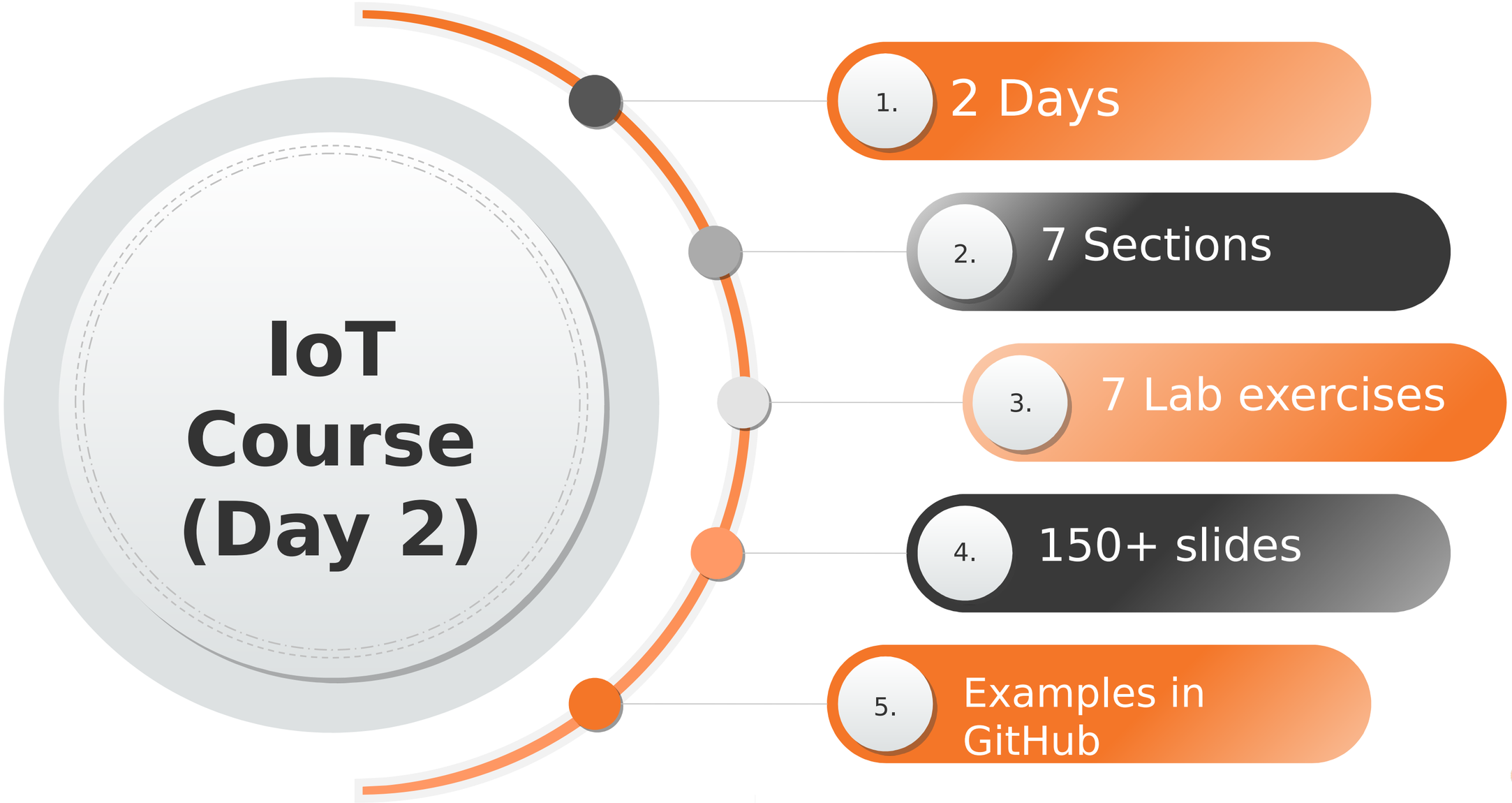


ANAVI Buttons in KiCad



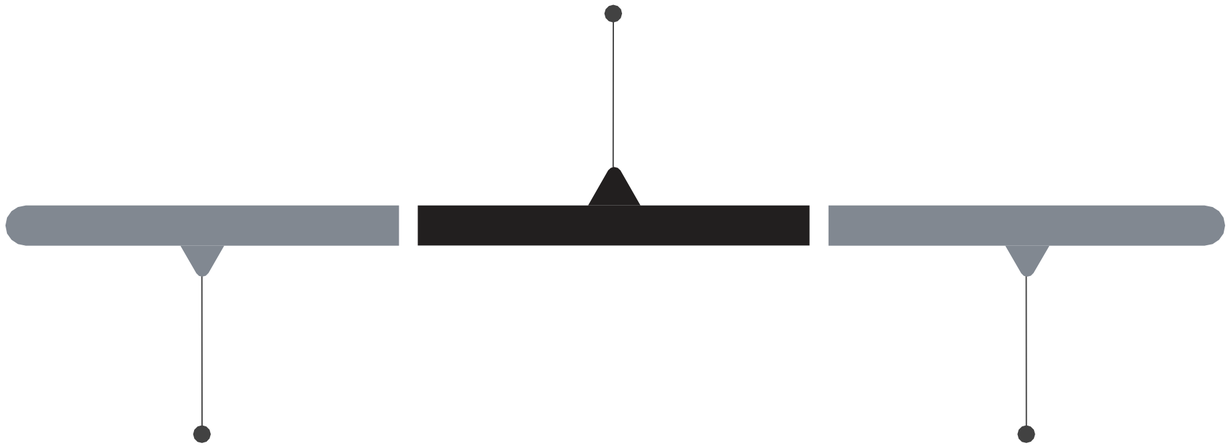


BEGINNING IOT WITH JAVASCRIPT
DAY 2


IoT Course (Day 2)
1. 2 Days
2. 7 Sections
3. 7 Lab exercises
4. 150+ slides
5. Examples in GitHub

Roadmap Day 2

MQTT



Sensor Modules

Integrating it all together!

Section 5: Sensors

In this lesson you will learn...

Finding sensors for Raspberry Pi

Exploring communication bus systems for controlling hardware peripherals attached to Raspberry Pi: 1-wire, I2C, I2S, SPI, UART, etc.

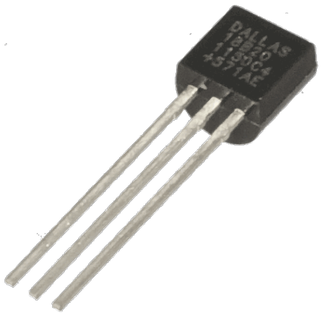
Deep diving in I2C communication bus system

Exploring HTU21D I2C sensor module for temperature and humidity

1-wire

- Serial signaling protocol for device communication
- Power and data can be send over a single wire
- Bi-directional or half-duplex signal transfer
- Many devices can share the same bus, each device has an unique 64-bit serial number
- Designed by Dallas Semiconductor (later acquired by Maxim Integrated)

1-wire & Raspberry Pi



- 1-wire needs to be enabled in Raspbian (via raspi-config, the Raspberry Pi Configuration app or manually in config.txt)
- Typically devices are connected to GPIO4 (aka pin 7) of Raspberry Pi, however it is also possible to use a custom pin
- Discovered 1-wire devices are listed in directory **/sys/bus/w1/devices/**
- Widely used 1-wire sensor is **18B20** digital sensor temperature (also available in waterproof housing)

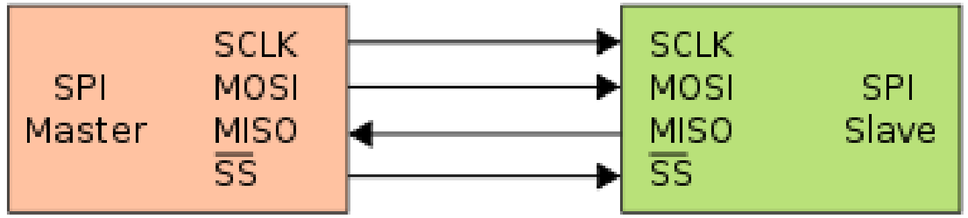
UART

- **U**niversal **a**synchronous **r**eceiver-**t**ransmitter (UART)
- Asynchronous serial communication protocol with rules for data, synchronization and parity bits as well support for various baud rates
- Standard baud rates are 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200
- Useful for connecting two devices together via serial communication, both devices must be configured to use the exact same protocol configurations

UART & Raspberry Pi

- Serial communication needs to be enabled in Raspbian (via raspi-config, the Raspberry Pi Configuration app or manually in config.txt)
- Pins 8 (GPIO14 UART0_TXD) and 10 (GPIO15 UART0_RXD) on Raspberry Pi are dedicated for serial communication
- 3.3V USB to UART debug adapters are often used for interfacing Raspberry Pi through the serial console
- More details: https://www.raspberrypi.org/documentation/configuration/uart.md

SPI



- **S**erial **P**eripheral **I**nterface (SPI)
- Synchronous serial communication interface specification used for short distance communication, often used for attaching displays to Raspberry Pi
- SPI bus on Raspberry Pi 40 pin header:

MOSI	**19**		
MISO	**21**		
SCLK	**23**	**24**	**CE0**
GND	**25**	**26**	**CE1**

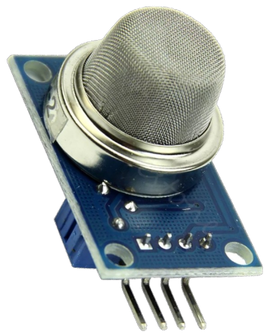
Analog sensors

- Raspberry Pi does **NOT** have a built-in analog to digital (A/D converter

- Popular external A/D converters:

Microchip MCP3002 dual channel A/D converter

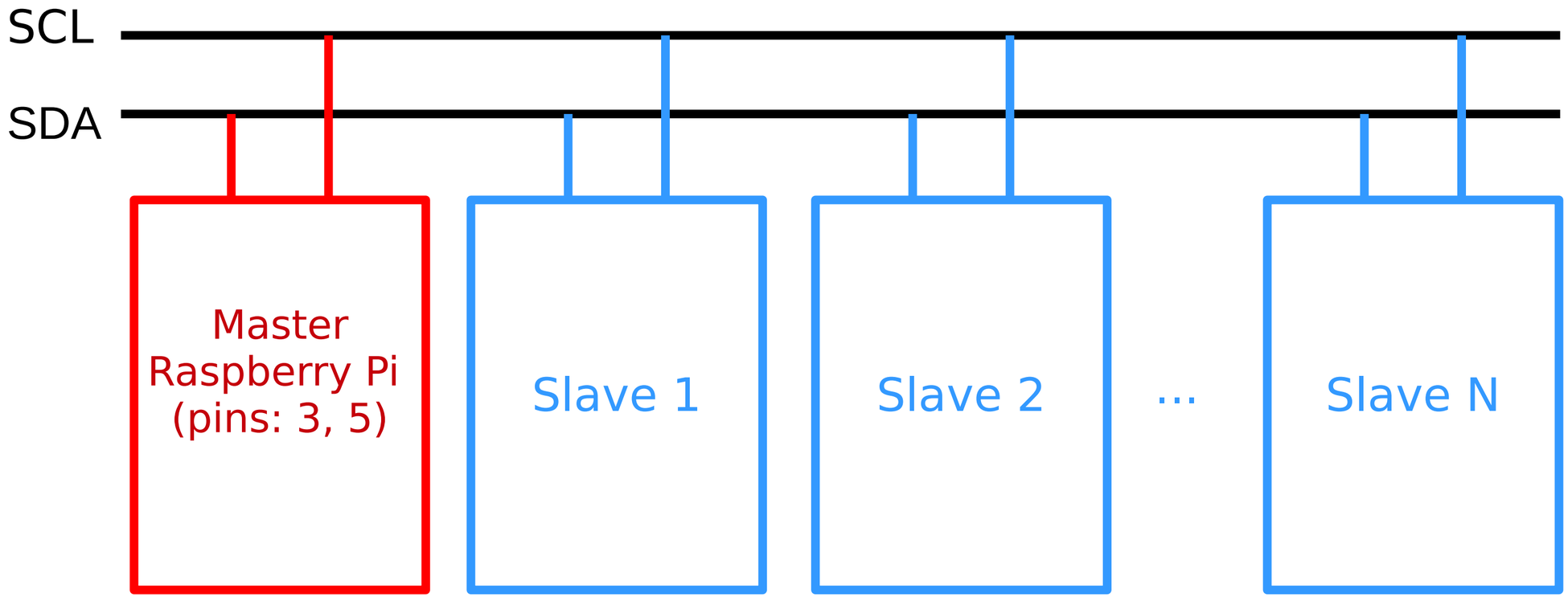
Microchip MCP3008 8 channel A/D converter



I2C

- **I**nter-**I**ntegrated **C**ircuit (I2C), pronounced I-squared-C
- Invented in 1982 by Philips Semiconductor (now NXP)
- Multi-master, multi-slave serial computer bus
- Each I2C slave type has an unique address, for 7-bit address up to 127 slaves can be attached
- No licensing fees to implement since 2006, fees are required for allocation of I2C slave address

I2C



I2C & Raspberry Pi

- I2C needs to be enabled in Raspbian (via raspi-config, the Raspberry Pi Configuration app or manually in config.txt)
- Pins 3 (GPIO2 SDA1) and 5 (GPIO3 SCL1) on Raspberry Pi are dedicated for serial communication
- Secondary I2C bus exists on pins 27 and 28 in any model and version of Raspberry Pi with 40 pin header but it dedicated only for reading device tree binary overlays from EEPROM (used in HAT)

Popular I2C Sensor Modules

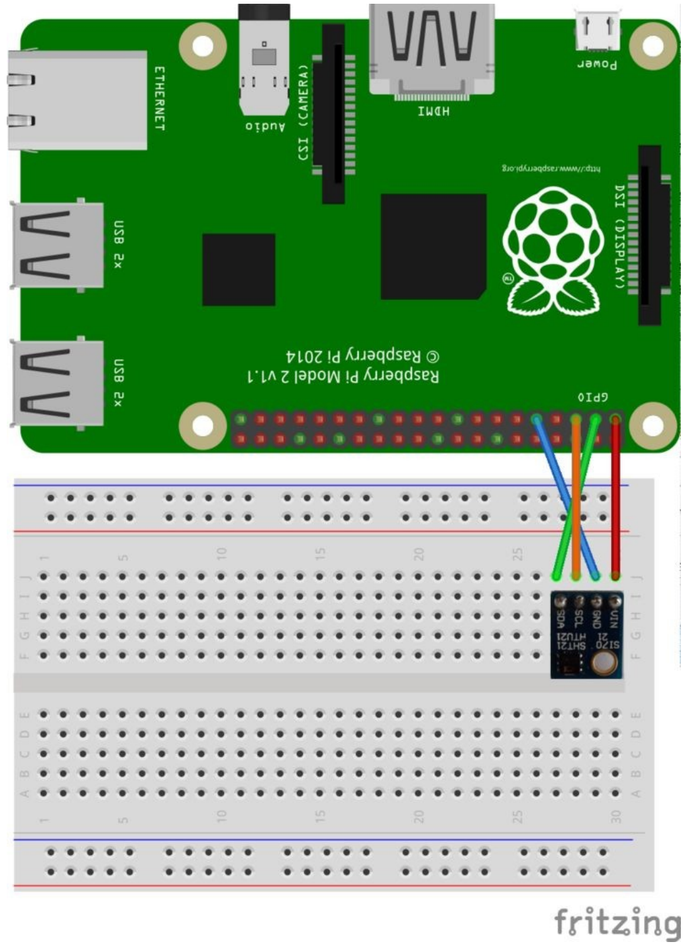
- HTU21D for temperature and humidity
- LMA75A for temperature
- BMP180 and BMP280 for temperature and barometric pressure
- BME280 for temperature, barometric pressure and humidity
- BH1750 or TSL2561 for light
- APDS-9960 for gesture and color detection
- Various character LED and OLED displays
- Many more...

HTU21D

- I2C sensor for measuring temperature and humidity by Measurement Specialties
- Temperature accuracy ±0.3C, humidity accuracy ±3%
- Operating temperature: -40C to 125C
- 7-bit I2C device address 0x40 (same as for **SI7021**)
- Available as a convenient module with breakout board from Adafruit, Sparkfun and other vendors

Wiring HTU21D I2C sensor module to Raspberry Pi

- Connect VCC to 3.3V (pin 1) on Raspberry Pi
- Connect GND to any of the ground pins of Raspberry Pi (pin 6, 9, 14, 20, 25, 30, 34, 39)
- Connect SDA to pin 3 of Raspberry Pi
- Connect SCL to pin 4 of Raspberry Pi



Reading temperature and humidity from HTU21D on Raspberry Pi with Node.js

```
var i2c = require('htu21d-i2c');
var htu21d = new i2c();
htu21d.readTemperature(function (temp) {
  console.log('Temperature: '+temp+'C');
  htu21d.readHumidity(function (humidity) {
    console.log('Humidity: '+humidity+'%');
  });
});
```

Slide10

In this lesson you learned...

- Variety of compute buses for attaching peripherals to Raspberry Pi
- Popular I2C sensor modules for Raspberry Pi
- Reading temperature and humidity from HTU21D I2C sensor module

Questions?

Lab 5: Reading temperature from HT21D with Node.js

Scenario:

Attach HTU21D I2C sensor module to Raspberry Pi
Read temperature and humidity using Node.js command-line application

Aim:

Learn how to interact with HTU21D I2C sensor module with Node.js and JavaScript

Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card with Raspbian
- HTU21D sensor module with female to female jumper wires

Steps

1) Create directory **lab5-htu21d** and file **htu21d.js** with the following content as in https://github.com/leon-anavi/iot-javascript-rpi/blob/master/lab5-htu21d/htu21d.js

2) Create **README.md** and add brief text description of the project

3) Run **npm init** and follow the on-screen instructions to create **package.json**

4) Add the following line to package.json to add the executable to the global PATH of the system:
"bin": "./htu21d.js",

5) Add the following line to list dependencies in package.json:

```
"dependencies": {
   "htu21d-i2c": ">=0.1.0"
},
```

6)Install the application:
sudo npm install -g

Demo

- Attach HTU21D I2C sensor module to Raspberry Pi
- Run the application:
  lab5-htu21d
- Verify that the output contains accurate information about temperature and humidity
- Heat the sensor by blowing hot air (or by just holding it in your arm for a while because the human body temperature is greater that the average room temperature)
- Run the application again and verify that the sensor is showing a greater temperature

Section 6: MQTT

In this lesson you will learn...

Exploring messaging protocols for real-time communication between Internet of Things

Introducing MQTT (messages, brokers, topics, wildcards, QoS, security, etc.)

Exploring open source MQTT brokers

Implementing MQTT client in Node.js

Popular Internet messaging protocols for Internet of Things (IoT)

- Message Queue Telemetry Transport (MQTT)
- MQTT for Sensor Networks (MQTT-SN)
- Constrained Application Protocol (CoAP)
- Advanced Message Queuing Protocol (AMQP)
- Data Distribution Service (DDS)

What is MQTT?

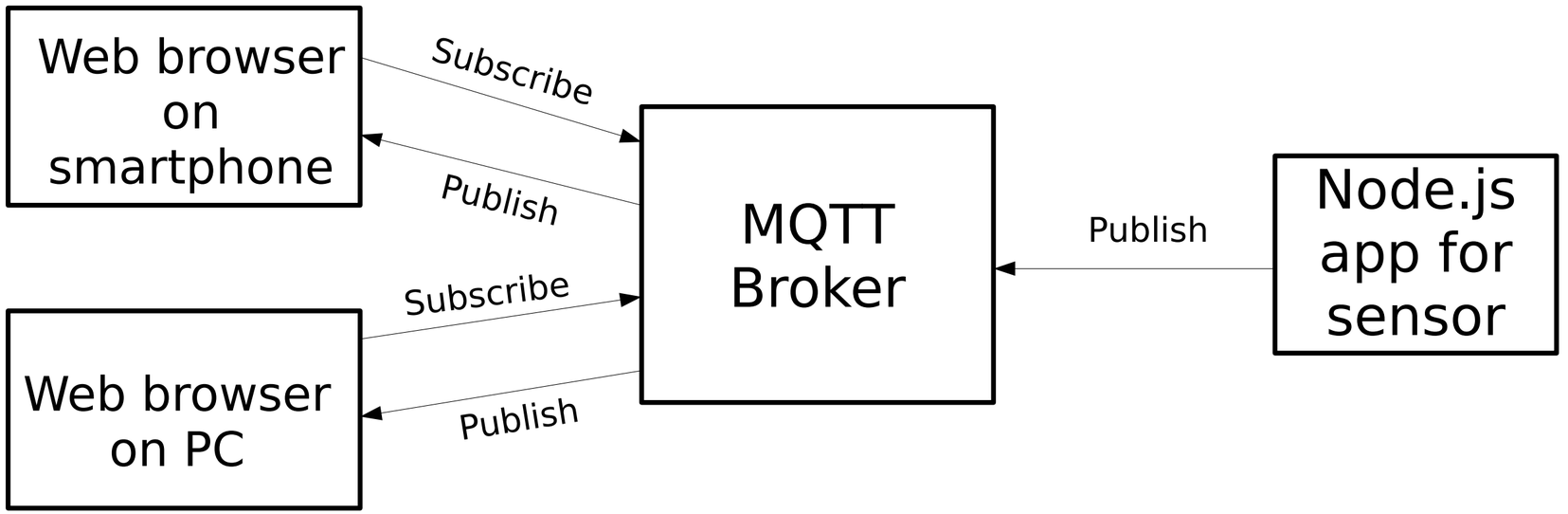
- Lightweight publish/subscribe machine-to-machine protocol on top of TCP/IP
- Near real-time communication
- Message broker
- Small source code footprint for embedded devices
- ISO standard (ISO/IEC 20922)

MQTT Operations

- Connect
- Disconnect
- Subscribe
- Unsubscribe
- Publish

Example usage

- Web page, loaded on web browsers, connect as MQTT clients via web sockets and subscribe to a topic
- Node.js application reads data from a sensor and publishes message which the MQTT broker delivers to the subscribed clients



MQTT Message

- Topic
- Payload (text or binary)
- Quality of service (0, 1 or 2)
- Retain (true or false)
- Furthermore, the MQTT protocol supports Last Will & Testament (LWT) which allows the broker to notify interested clients about an ungracefully disconnected client by publishing a message on his behalf

Example message	
Topic	"hello/1"
Payload	{ "temperature": 20 }
QoS	2
Retain	false

MQTT Topics and Wildcards

- Topic

  home / bedroom / temperature

- Single level wildcards

  home / + / temperature

- Multiple levels wildcards

  home / #

MQTT Security

- Transport encryption with TLS/SSL
- Authentication: username/password
- Authorization: Access Control Lists (ACL)

Open Source MQTT Brokers

- Mosquitto
- HiveMQ (commercial license, open source plugins)
- Mosca
- emqttd
- VerneMQ
- ActiveMQ
- RabbitMQ

Mosquitto

- Free and open source MQTT broker implement in the C programming language
- Supports MQTT protocol version 3.1 and 3.1.1
- Available for all popular GNU/Linux distributions: Debian,Ubuntu, Fedora, RedHat Enterprise Linux, openSUSE, CentOS, ArchLinux, Slackware, etc.
- Also available for Windows, FreeBSD and Mac
- Project of iot.eclipse.com

Public MQTT Brokers

Server	Broker	Port	Websocket
iot.eclipse.org	Mosquitto	1883 / 8883	80 / 443
test.mosquitto.org	Mosquitto	1883 / 8883 / 8884	8080 / 8081
test.mosca.io	Mosca	1883	80
broker.hivemq.com	HiveMQ	1883	8000
broker.mqttdashboard.com	HiveMQ	1883	8000
cloudmqtt.com	Mosquitto	1xxxx	3xxxx

Popular Open Source libraries for MQTT Clients

- Paho offers MQTT client implementations for C/C++, Java, JavaScript, Python and C#
  http://www.eclipse.org/paho/
- Node.js library for implementing MQTT clients
  https://www.npmjs.com/package/mqtt
- Arduino client for MQTT:
  http://pubsubclient.knolleary.net/

Using MQTT in Node.js (1/2)

- MQTT.js is a client library for the MQTT protocol available through **npm**: https://www.npmjs.com/package/mqtt

- Connect to MQTT broker:
```
var mqtt = require('mqtt');
var client  = mqtt.connect('mqtt://test.mosquitto.org');
```

- Publish message:
```
client.publish('home/room', 'hello');
```

Using MQTT in Node.js (2/2)

- Subscribe for a topic:

```
client.on('connect', function () {
   client.subscribe('home/#', function (err) {
      //Do something...
   })
});
```

- Handle event **message** to process a received message:

```
client.on('message', function (topic, message) {
   // Do something with the received message
});
```

Slide10

In this lesson you learned...

- Popular messaging protocols for IoT
- Deep dive in MQTT
- Mosquitto and other open source MQTT brokers
- Using MQTT with JavaScript in Node.js

Questions?

Lab 6: Getting started with MQTT on Raspberry Pi

Scenario:

Install Mosquitto MQTT broker on Raspberry Pi
Verify that the broker is working by subscribing and publishing MQTT messages

Aim:

Install MQTT broker and understand how MQTT works

Steps

1) Open a terminal and type the following commands to install Mosquitto
sudo apt update
sudo apt install -y mosquitto mosquitto-clients
2) Enable automatic launch of Mosquitto at system start-up:
sudo systemctl enable mosquitto.service
3) Verify that Mosquitto has been installed:
mosquitto -h
4) Subscribe for a topic:
mosquitto_sub -h localhost -d -p 1883 -t "hello"
5) Open another terminal and publish a message:
mosquitto_pub -h localhost -d -p 1883 -t "hello" -m "test"
6) In the first terminal, verify that the message has been received

One more thing...

7) To enable web socket support in your MQTT broker, create **/etc/mosquitto/conf.d/websocket.conf** (requires **sudo**) and add the following lines:
listener 1883
listener 1884
protocol websockets

8) Restart mosquitto:
sudo systemctl restart mosquitto

9) Verify again that mosquitto is running successfully by repeating steps 3 to 5

BREAK

10min

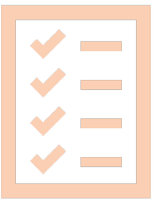
Section 7: Web Sockets

In this lesson you will learn...

Introducing the Paho Project

Understanding how web sockets work

Receiving and sending data through MQTT over web sockets directly from a web browser



What is a WebSocket?

- Enables web applications to maintain bidirectional communications with server-side processes
- Full-duplex communication channel over a single TCP connection
- Defined by a Web sockets specification in HTML5: https://html.spec.whatwg.org/multipage/web-sockets.html
- Supported by major and most popular web browsers: Google Chrome, Firefox, Safari, Opera, Microsoft Edge and Internet Explorer

Why WebSocket is important for MQTT?

- Hypertext Transfer Protocol (HTTP) defines session as a sequence of network request-response transactions which is **not** convenient for real-time events
- MQTT provides (near) real-time communication, often based on real-life events such as change of temperature, humidity, detection of movement, power consumption, etc.
- WebSocket allows quick and convenient way to process quickly MQTT messages and to display them to in user-friendly way to the user through modern HTML5 technologies

The Paho Project

- MQTT publish/subscribe client libraries for popular programming languages such as C, C++, Java, Python, Go, Rust, C# and JavaScript
- Free and open source available under Eclipse Distribution License 1.0 (BSD) and Eclipse Public License 1.0
- Actively developer by IBM and a lot of individual contributors
- Project of iot.eclipse.com
- Part of other popular Eclipse projects such as Photon, Oxygen, Neon, and Luna
- https://www.eclipse.org/paho/

Eclipse Paho JavaScript Client

- MQTT client library for easy integration in HTML5 web pages
- Written in JavaScript
- Uses WebSockets to connect to an MQTT Broker
- Documentation: http://www.eclipse.org/paho/files/jsdoc/index.html

Eclipse Paho JavaScript Client Features

MQTT 3.1	✔
MQTT 3.1.1	✔
LWT	✔
SSL / TLS	✔
Message Persistence	✔
Automatic Reconnect	✔

Offline Buffering	✖
WebSocket Support	✔
Standard TCP Support	✖
Non-Blocking API	✔
Blocking API	✖
High Availability	✔

Eclipse Paho JavaScript Client CDNs

- Include the following line to use the plain library:
  <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>

- Include the following line to use the minified library:
  <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js" type="text/javascript"></script>

Class Message Properties

- **payloadString** (string) the payload as text with valid UTF-8 characters.
- **payloadBytes** (ArrayBuffer) the payload as an ArrayBuffer.
- **destinationName** (string) mandatory property with the topic of the MQTT message
- **qos** (number) represents MQTT QoS, must be 0, 1 or 2, the default value is 0
- **retained (**Boolean) specified if the message has to be retained by the MQTT broker, if true it will be delivered to any new subscriber for this topic
- **duplicate** (Boolean) true if the message might have been already received, only set on messages received from the server

Using WebSocket for MQTT in Web Page (1/3)

- Create MQTT client:
  client = new Paho.MQTT.Client("iot.eclipse.org", 443, "myMqttClient");

- Assign callback functions for processing received MQTT messages:
  client.onMessageArrived = onMessageArrived;

- Connect to MQTT broker:
  client.connect({onSuccess:onConnect});

Using WebSocket for MQTT in Web Page (2/3)

- Subscribe to MQTT topic after successfully establishing a connection with the MQTT broker:

```
function onConnect() {
    client.subscribe("home/#");
}
```

- Process received messages in the callback function assigned previously:

```
function onMessageArrived(message){
    //Do something with the message
}
```

Using WebSocket for MQTT in Web Page (3/3)

- Publish MQTT messages using the JavaScript library from the Paho project:

```
message = new Paho.MQTT.Message("Hello");
message.destinationName = "World";
client.send(message);
```

Slide10

In this lesson you learned...

- Using WebSockets
- Using the Paho project for developing MQTT client in a HTML5 web page

Questions?

Lab 7: Integrating it all together

Scenario:

Read temperature and humidity from HTU21D I2C sensor module and publish it to MQTT broker from Node.js command-line application
Create a web page that connects to MQTT broker through web socket using the Paho library, subscribe for receiving message and show temperature and humidity, publish MQTT messages to change colors of the traffic light

Aim:

Integrate everything together and learn how to use MQTT

Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card with Raspbian
- HTU21D sensor module with female to female jumper wires
- ANAVI Traffic Lights (or a breadboard with jumper wires, 3 resistors and red, yellow and green 5mm LED)

Steps

1) Create directory **lab7-mqtt** and file **mqtt-sensor.js** with the following https://github.com/leon-anavi/iot-javascript-rpi/blob/master/lab7-mqtt/mqtt-sensor.js
2) Create **README.md** and add brief text description of the project
3) Run **npm init** and follow the on-screen instructions to create **package.json**
4) Add the following line to package.json to add the executable to the global PATH of the system:
"bin": "./mqtt-sensor.js",
5) Add the following line to list dependencies in package.json:

```
"dependencies": {
    "htu21d-i2c": ">=0.1.0",
    "mqtt": ">=2.18.8",
    "onoff": ">=3.2.1"
},
```

6) Install the application:
sudo npm install -g
7) Create HTML5 web page with Paho to read data from the sensor, turn on and off the traffic lights

Demo

- Attach HTU21D I2C sensor module and ANAVI traffic lights to Raspberry Pi
- Run the application: **lab7-mqtt**
- Verify that the MQTT broker is running
- Open a web browser, load the HTML5 web page and verify that the output contains accurate information about temperature and humidity
- Verify that you can turn on and off the lights through MQTT

… and one more thing

- Adafruit http://adafruit.com/
- Sparkfun https://www.sparkfun.com/
- Olimex https://www.olimex.com/
- Tindie https://www.tindie.com/
- Crowd Supply https://www.crowdsupply.com/

Further reading

- Raspberry Pi Zero W Wireless ProjectsRaspberry Pi Zero W Wireless Projects
  https://www.packtpub.com/hardware-and-creative/raspberry-pi-zero-w-wireless-projects

- Node.js By Example, Krasimir Tsonev
  https://www.packtpub.com/application-development/nodejs-example

- Embedded Linux Projects Using Yocto Project Cookbook, Alex González
  https://www.packtpub.com/virtualization-and-cloud/embedded-linux-projects-using-yocto-project-cookbook

- Yocto for Raspberry Pi, Pierre-Jean Texier, Petter Mabäcke
  https://www.packtpub.com/hardware-and-creative/yocto-raspberry-pi

THANK YOU!