Raspberry Pi with Java

a) **Intro to Raspberry Pi and Java ME 8**
b) **Raspberry Pi Tricks**

#RPiJava8
#DevNexus

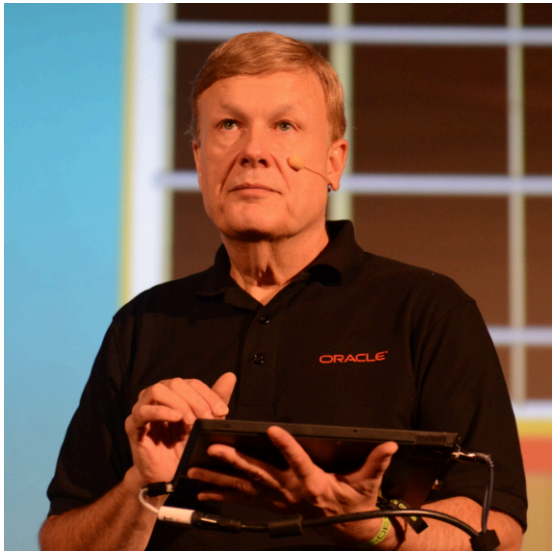
James Weaver @JavaFXpert
Stephen Chin @SteveOnJava
Java/IoT/Cloud Technology Ambassadors
Oracle Corporation

Based loosely on upcoming “Raspberry Pi with Java” book
by @SteveOnJava and @JavaFXpert

#RPiJava
#DevNexus

About the first presenter

#RPiJava
#DevNexus



James Weaver

Java/IoT/Cloud Technology Ambassador
Oracle Corporation
Twitter: @JavaFXpert
Email: ***james***.weaver@oracle.com

Presentation based upon freely available MOOC

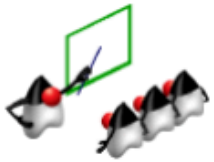
ORACLE Learning Library

Home | Products | Search | My Library

Oracle Course: Develop Java Embedded Applications Using a Raspberry Pi

Tom McGinn, Angela Caicedo, Jim Weaver, Simon Ritter

Welcome | Lessons



Welcome

Welcome to the Developing Java Embedded Application Open Online Course!

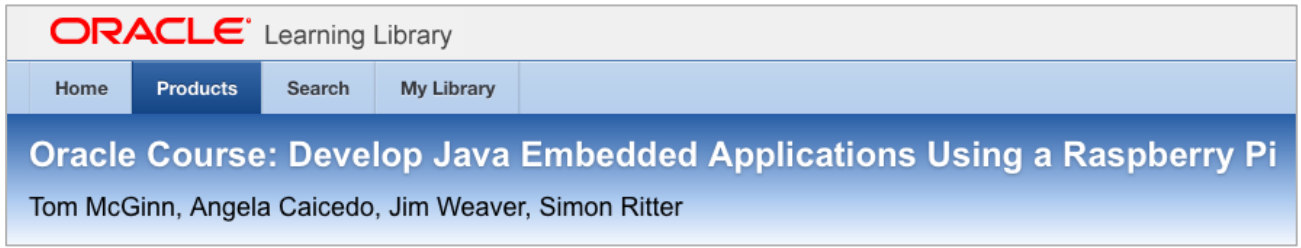
- In the Course Announcements section, please take a few minutes to read the course outline. This document describes what you can expect from this course and how it differs from traditional classroom training.
- The Required Hardware section includes a link to a document that describes the hardware requirements for the course and links to two options for purchasing the hardware.
- Please take a moment to join the forum - you'll meet your fellow students there! **The course materials are now open to the public.**

MOOC has videos, code, and other resources

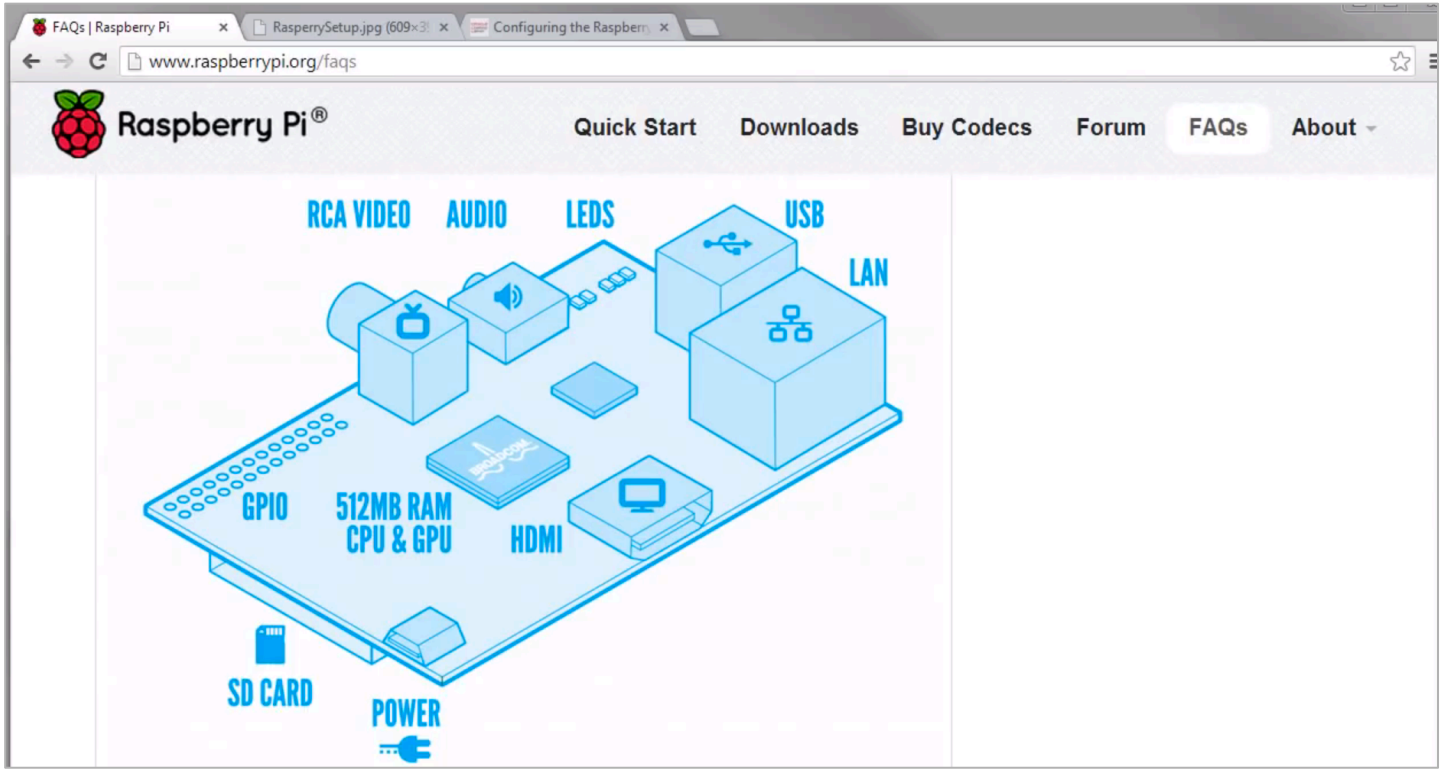
Introducing the Raspberry Pi as an Embedded Device
Course Overview (5:14)
Installing the Java ME 8 SDK and NetBeans 8 IDE…
Tutorial: Installing the Raspberry Pi Software
Unboxing the Adafruit Accessory Pack (6:12)
Soldering the BMP and GPS Sensor Breakouts (7:…
Lesson 1-1: Introduction to the Raspberry Pi (3:01)
Lesson 1-2: Introduction to Java ME Embedded (…

Supporting Materials
Java MEEP 8 API java…
CLDC 8 API javadoc
Device I/O API for Jav…
GCF API javadoc
Java ME Embedded D…
NetBeans Java Develo…
Lesson 1 Homework Zi…

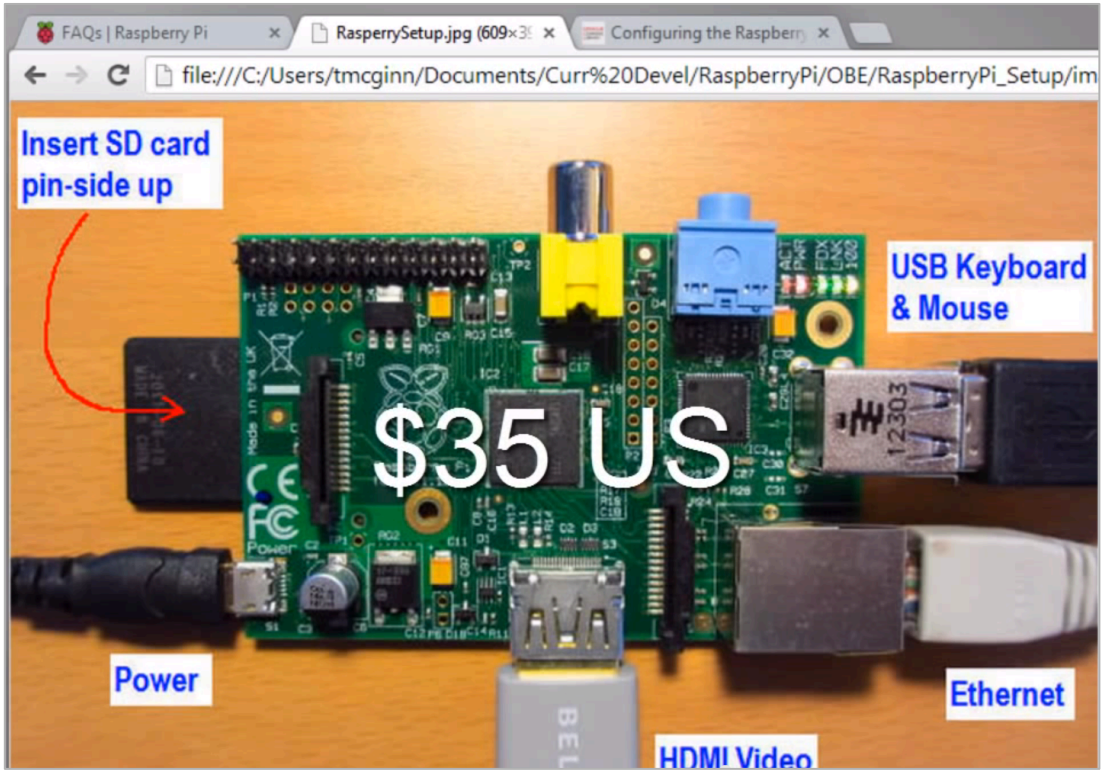
Tweet contains Java Embedded Rasp Pi MOOC



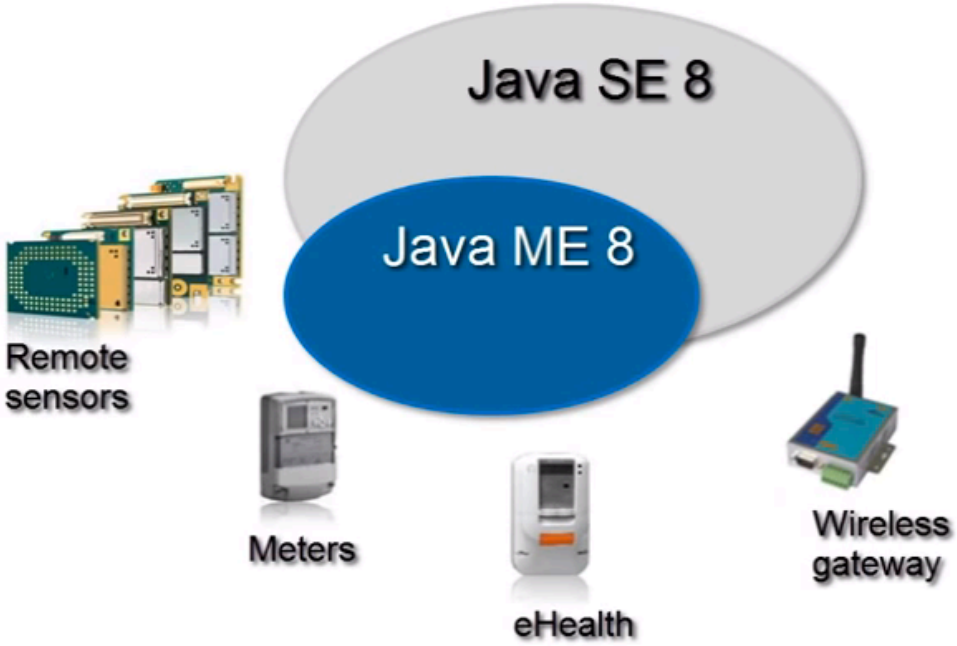
Raspberry Pi Hardware Components



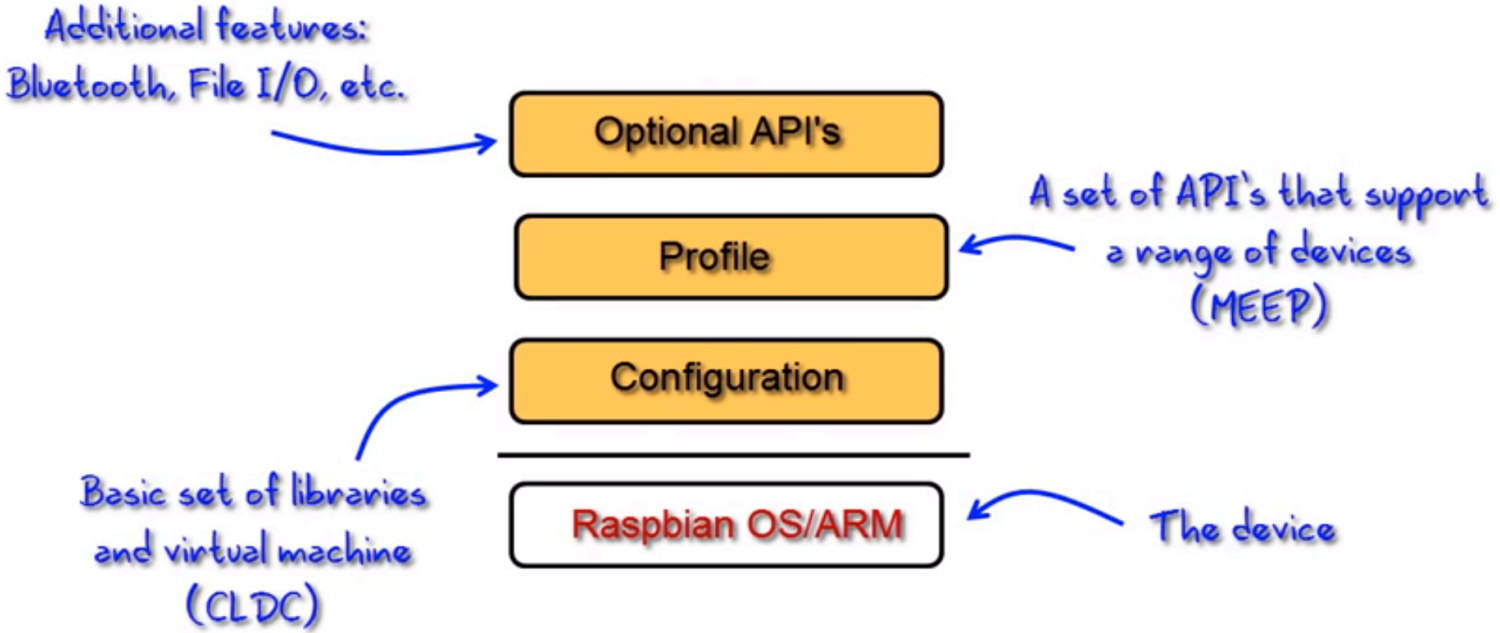
Raspberry Pi Hardware Setup



Java SE vs. Java ME



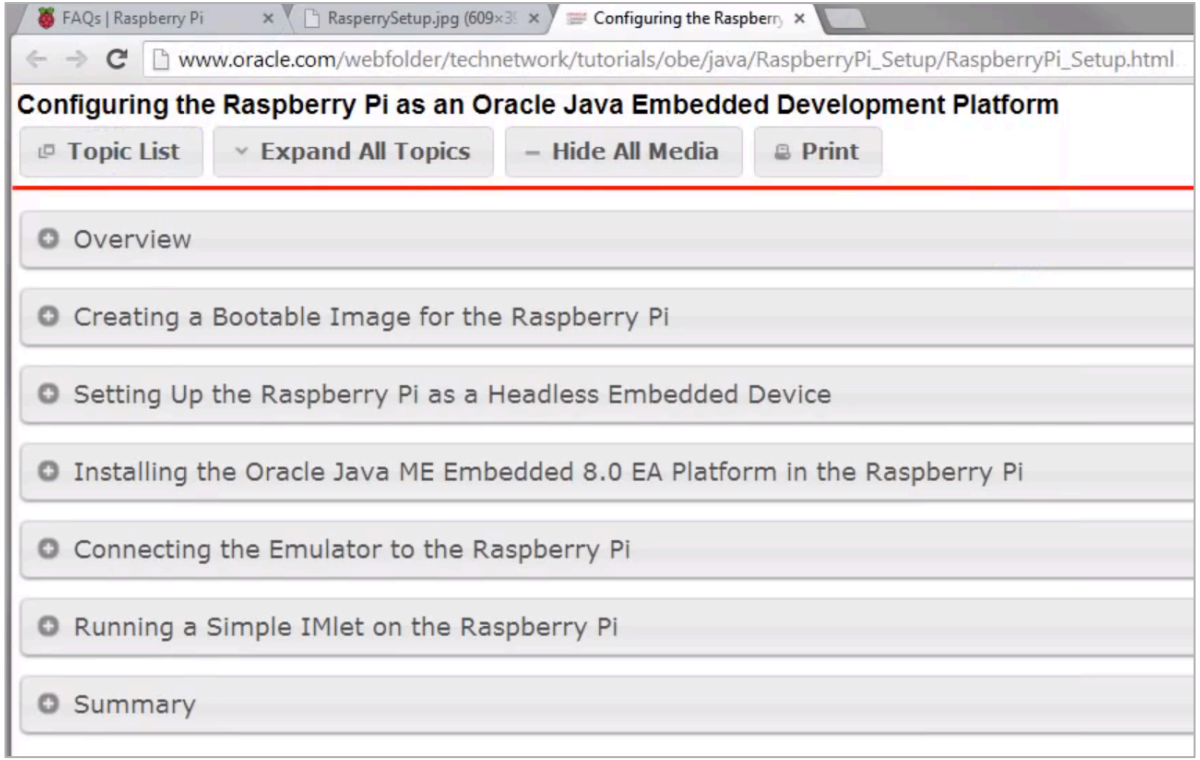
Java ME 8 Architecture



Oracle MOOC: Software Installation Instructions On Your Windows PC

1. Install Java SE JDK 8
2. Install Java ME SDK 8
3. Install NetBeans 8 IDE
4. Install NetBeans ME SDK Plugins

Config the Raspberry Pi for Java Development



Java ME Application Lifecycle Methods

```
public class SampleApp extends MIDlet {
    @Override
    public void startApp() {
        System.out.println("Starting...");
    }

    @Override
    public void destroyApp(boolean unconditional) {
        System.out.println("Destroying...");
    }
}
```

Our Project

... and Some Relevant Concepts Discussed

GPIO

Door open/close

I²C

Pressure/
Temperature

GPS, UART

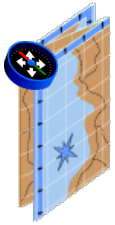
Location/
Altitude

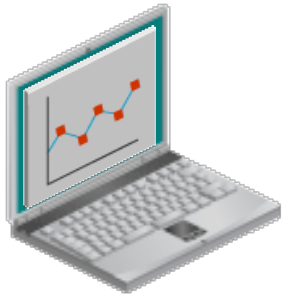
RMS

Storage

JavaFX

Client Display

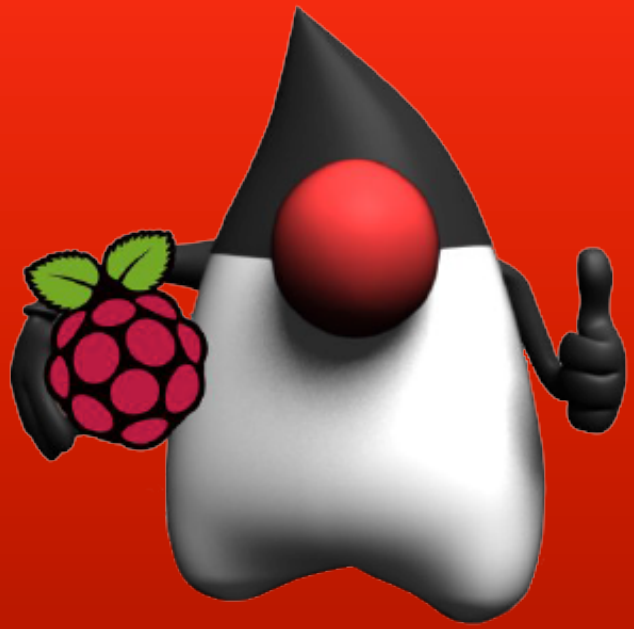




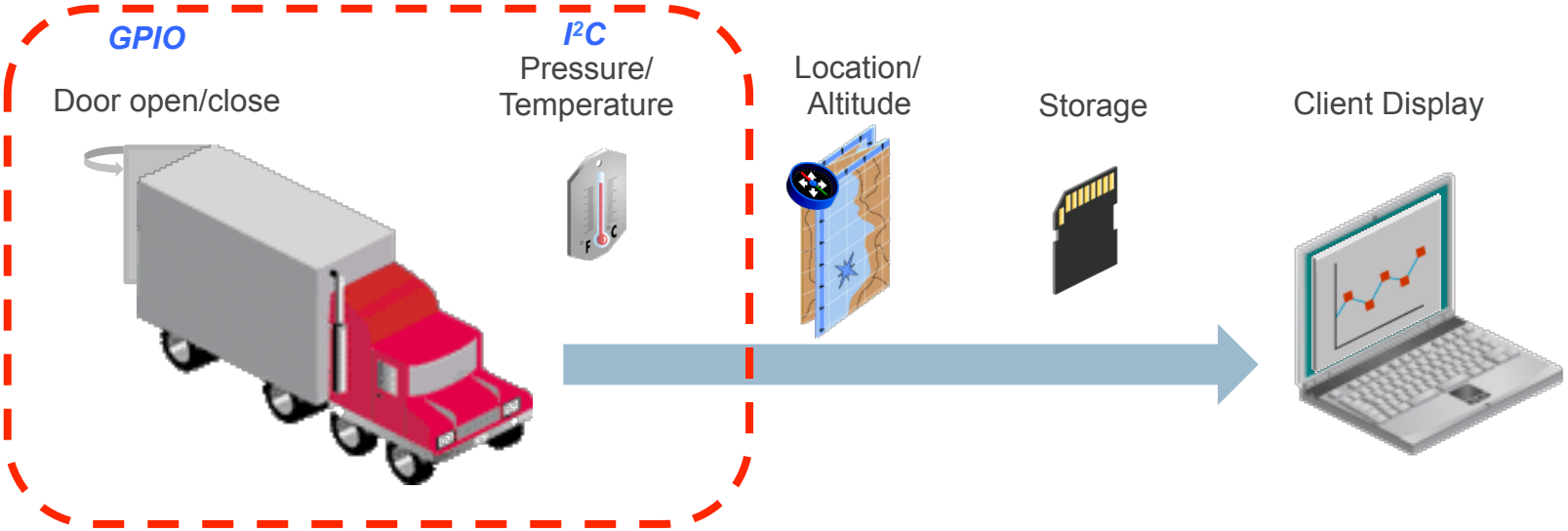
In this course, you will build a prototype of an embedded device to collect, store, analyze and share data from a shipping container.

ORACLE

Lesson 2:
Using the GPIO and I2C on the Raspberry Pi



In This Lesson…



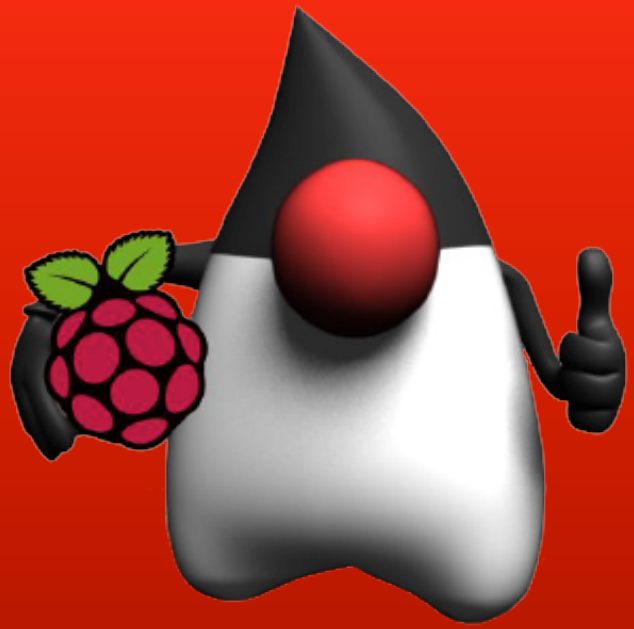
In this course, you will build a prototype of an embedded device to collect, store, analyze and share data from a shipping container.

Lesson 2 Agenda

- Getting started with the hardware
- Working with GPIO on the Raspberry Pi
- I^2C Overview
- Enabling I^2C on the Raspberry Pi
- Writing code for I^2C

ORACLE

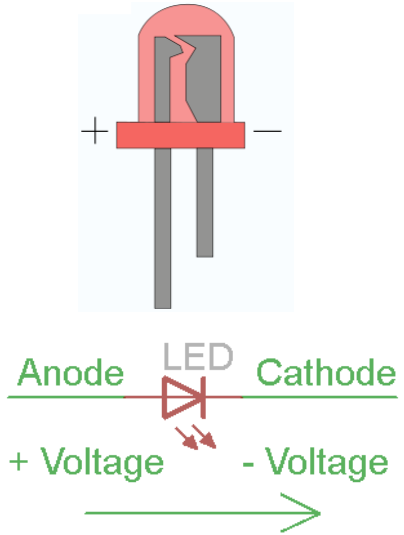
Lesson 2.1:
Raspberry Pi Cooking Class:
Lets see the Ingredients



Hardware

LED

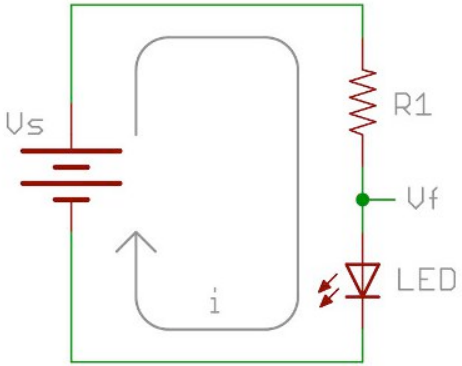
- Light Emitting Diode (LED): red, green, blue
- Two wires:
  - **Anode** positive, longest, and
  - **Cathode** negative, shortest



Hardware

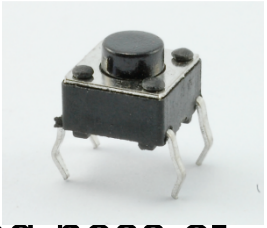
Resistors

- Their job is to “resist”, regulate or to set the flow of electrons (current) through them.
- Used with LED to keep the current at specific level: characteristic forward current
- 560Ω and 10KΩ

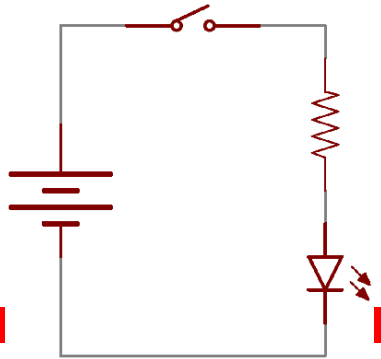


Hardware

Switch

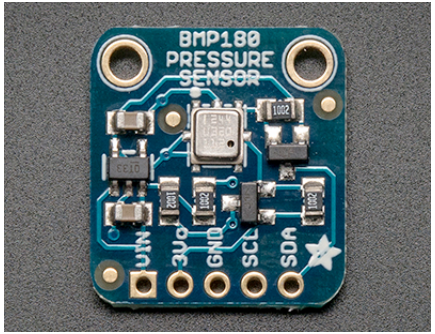


- Switch is a component which controls the open-ness or closed-ness of an electric circuit.
- Two states:
  - **Off**: looks like an open gap in the circuit (**open circuit)**, preventing current from flowing.
  - **On:** acts just like a piece of wire (**closes the circuit)**, turning the system “on” and allowing current to flow
- Momentary switch:
  - only **on** when the button is pressed.
  - you release the button, the circuit is broken.



Hardware

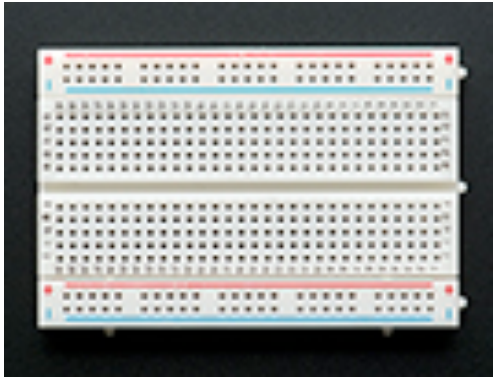
BMP180



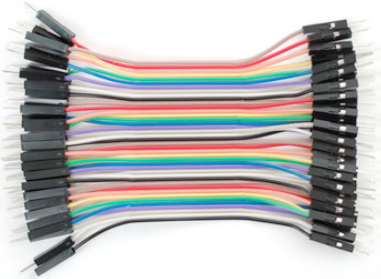
- Measure barometric pressure and temperature
- Specifications:
  - Pressure sensing range: 300-1100 hPa (9000m to -500m above sea level)
  - Up to 0.03hPa / 0.25m resolution
  - -40 to +85°C operational range, +-2°C temperature accuracy
  - 2-pin I^2C interface on chip
  - Uses 3.3-5V power and logic level for more flexible usage

Hardware

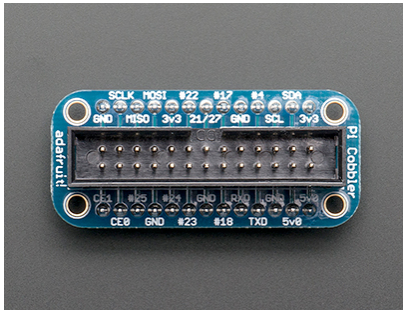
Connecting things…



Breadboard



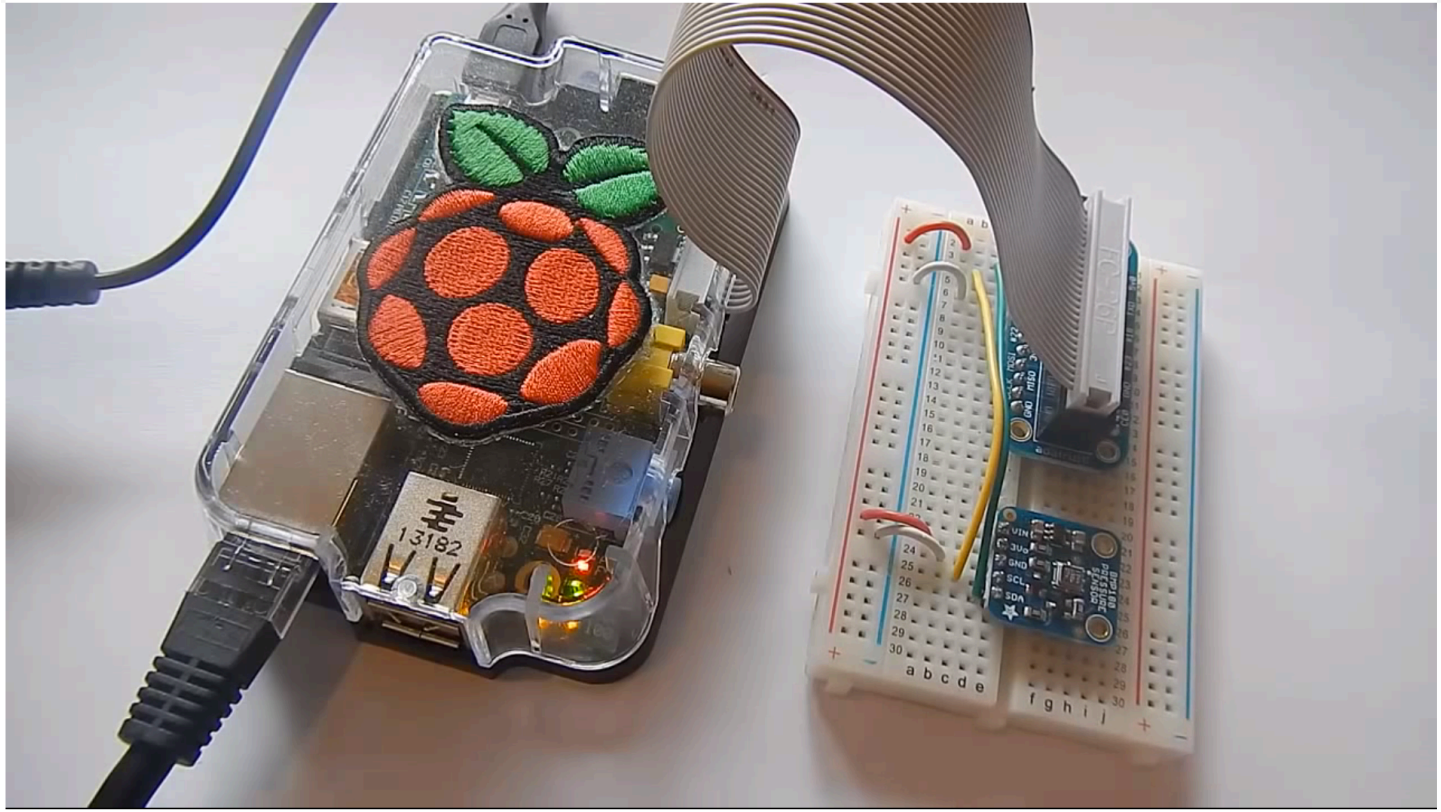
Jumper cables



Pi Cobbler Breakout

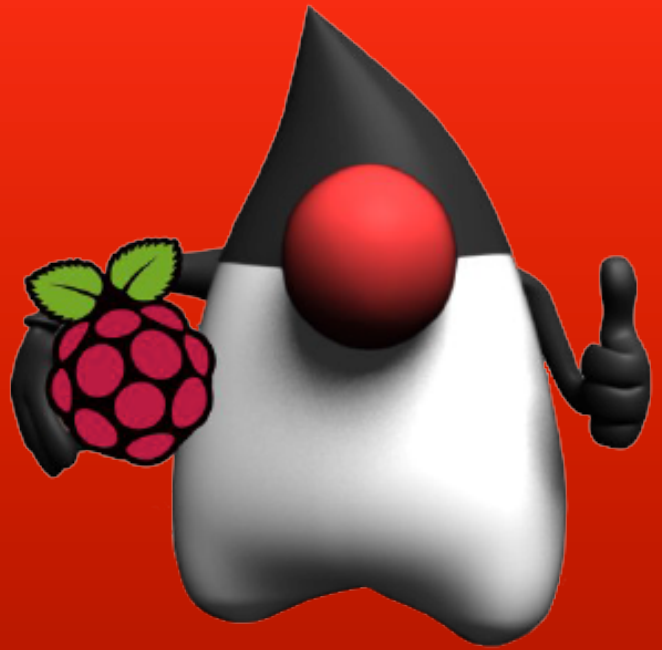
GPIO ribbon cable

Some hardware connected to the Raspberry Pi



ORACLE

Lesson 2.2:
Getting Started with GPIO



General Purpose Input/Output (GPIO)

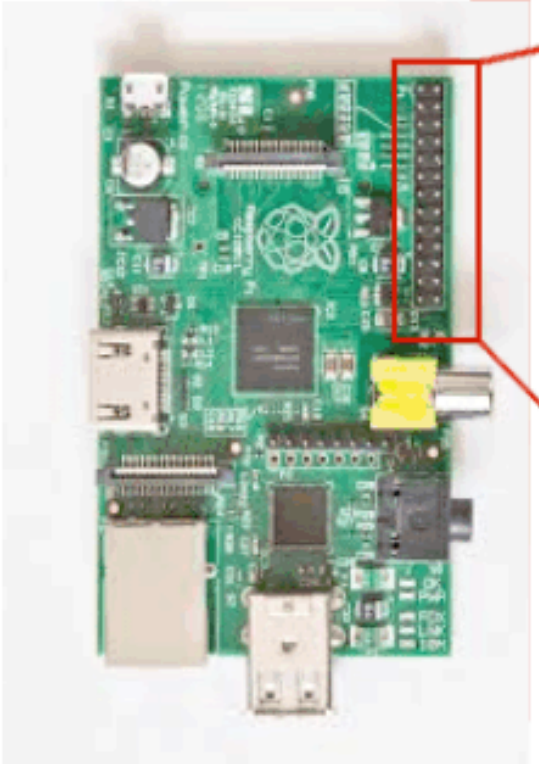
- GPIO pin is a generic pin with high or low
  - Its behavior can be programmed through software.
- GPIO port is a platform-defined grouping of GPIO pins (often 4 or more pins).

```
GPIOPin pin =  DeviceManager.open(1);
GPIOPort port =  DeviceManager.open(0);
pin.setValue(true);
port.setValue(0xFF)
```

GPIOPin Interface

- Provides methods for controlling a GPIO pin.
- A GPIO pin can be configured for output or input.
  - Output pins are both writable and readable
  - Input pins are only readable
  - **`GPIOPin.INPUT  GPIOPin.OUTPUT`**
- Get its value by polling, or register a **`PinListener`**
  - To register a **`PinListener`**: **`setInputListener(PinListener)`**
  - To remove the listener: **`setInputListener(null)`**
  - Writing a **`GPIOPin`** results in **`UnsupportedOperationException`**

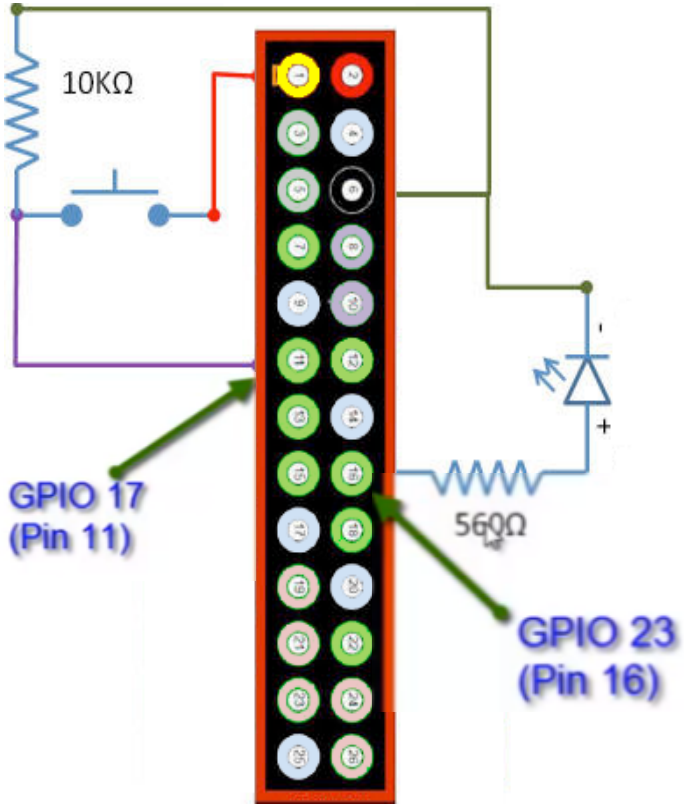
GPIO Pins on the Raspberry Pi



Broadcom names			
3.3v	1	2	5v
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	0v
GPIO 4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SPI MOSI	19	20	DNC
SPI MISO	21	22	GPIO 25
SPI SCLK	23	24	SP10 CEO N
DNC	25	26	SP10 CE1 N

Turning a LED ON and OFF

The Circuit



GPIO with Device Access API

```
public class GPIOTest implements PinListener {
...
 private static final int BTN_PIN = 17;
 private static final int BTN_PORT = 0;
 private static final int LED1_ID = 23;

 ...
 GPIOPinConfig config1 = new GPIOPinConfig(BTN_PORT,
                         BTN_PIN,
                         GPIOPinConfig.DIR_INPUT_ONLY,
                         DeviceConfig.DEFAULT,
                         GPIOPinConfig.TRIGGER_BOTH_EDGES,
                         false);

 GPIOPin button1 =  DeviceManager.open(config1);
 GPIOPin led1 = DeviceManager.open(LED1_ID);
 button1.setInputListener(this);
```

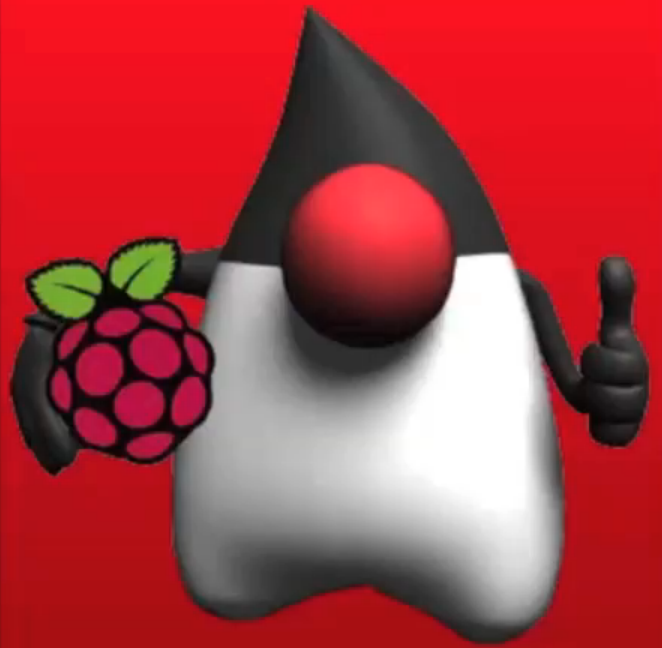
Trigger both edges causes a PinListener event when the state of the switch (button) rises from low (0) to high (1), or falls from high (1) to low (0).

GPIO with Device Access API

```
@Override
public void valueChanged(PinEvent event) {
    GPIOPin pin =  event.getDevice();
    try {
      if (pin == button1) {
          // Toggle the value of the led
          led1.setValue(!led1.getValue());
      }
    } catch (IOException ex) {
          System.out.println("IOException: " + ex);
      }
}
```

ORACLE

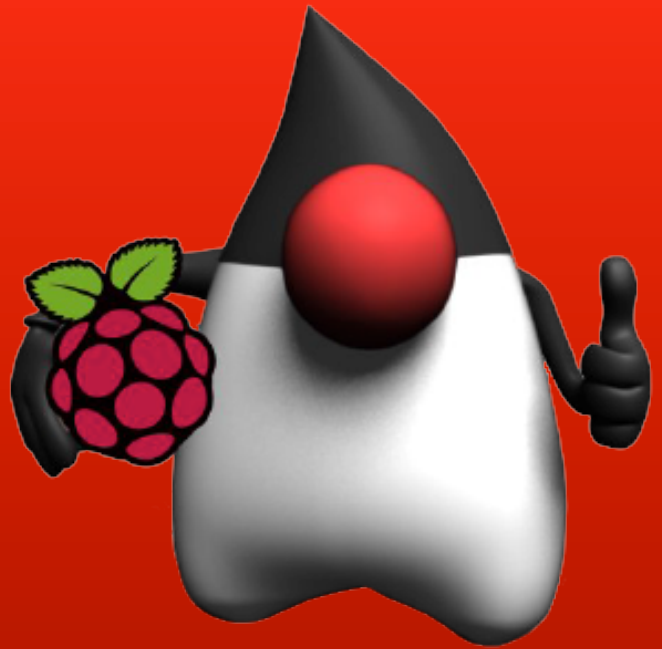
Exercise:
Turning an LED ON and OFF



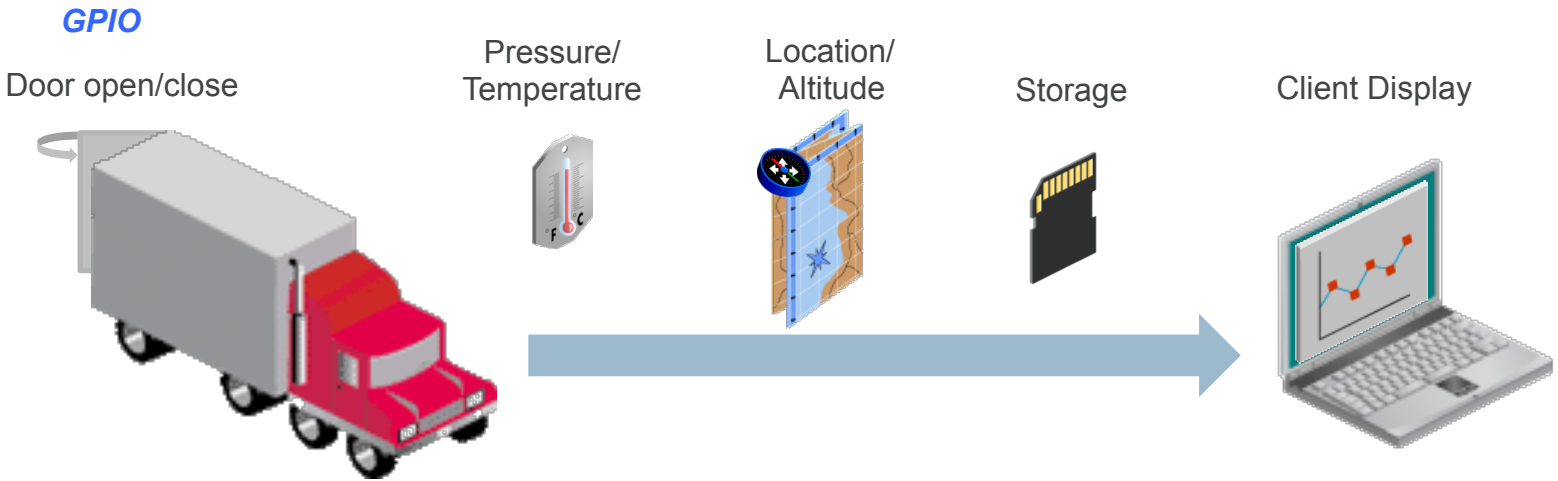
ORACLE

Homework:
Create a Door Sensor

Ingredients:
1 Green LED
1 Red LED
1 Momentary switch



Door open/close



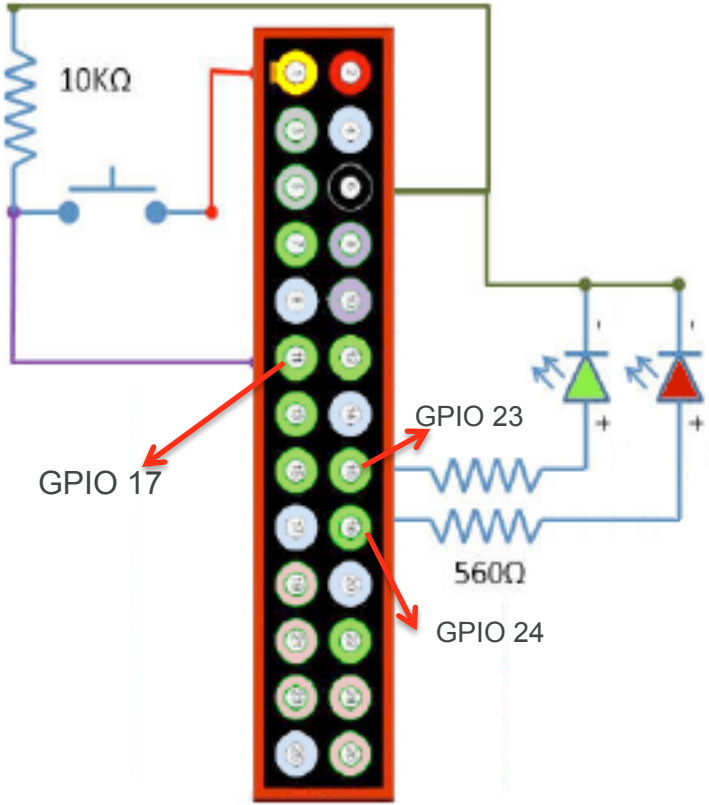
In this course, you will build a prototype of an embedded device to collect, store, analyze and share data from a shipping container.

The Door Sensor

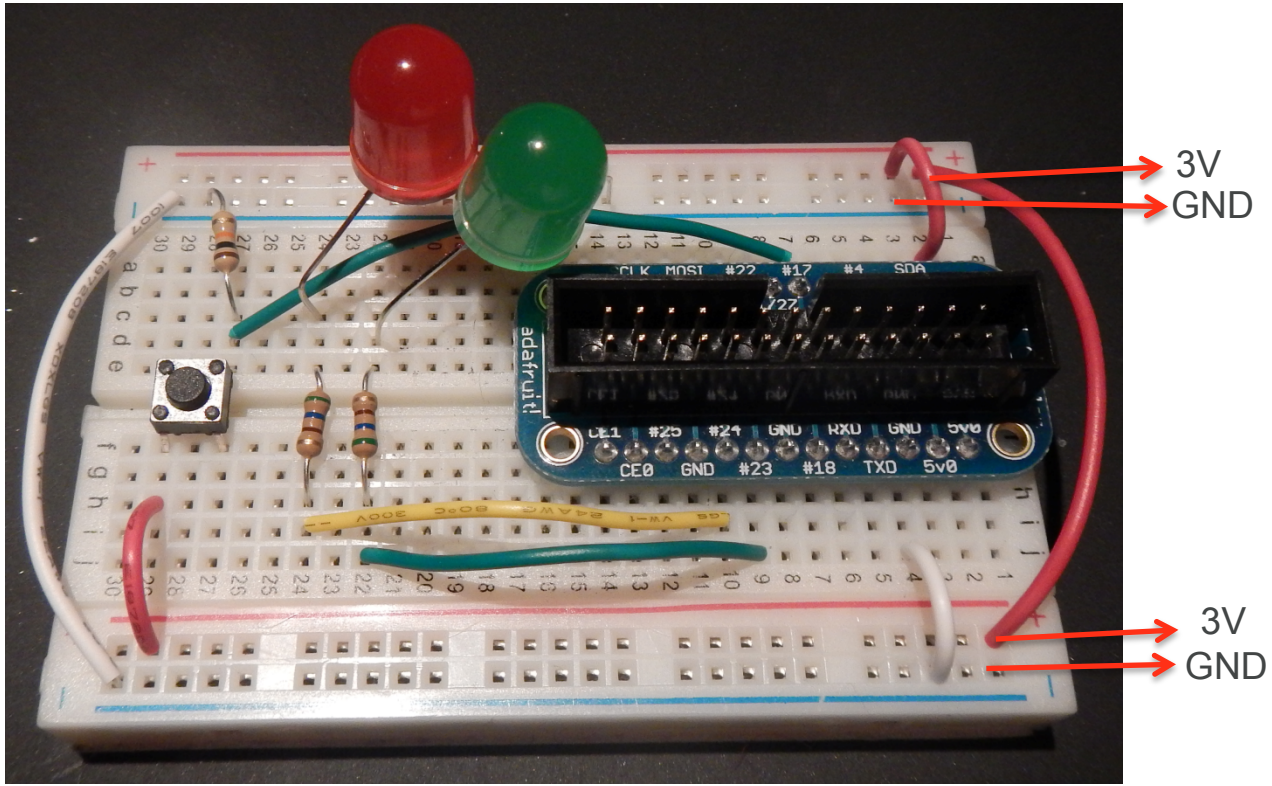
What we are trying to do

- The idea is to show the door status via LEDs.
  - If the door is closed, the packages are safe in the truck: green LED is ON
  - If the door is open, we should warn: red LED is ON, or blinking (your choice)
- Pins used in the solution (not required to be the same)
  - Green LED in pin 23
  - Red LED in pin 24
  - Switch in pin 17

The Wiring

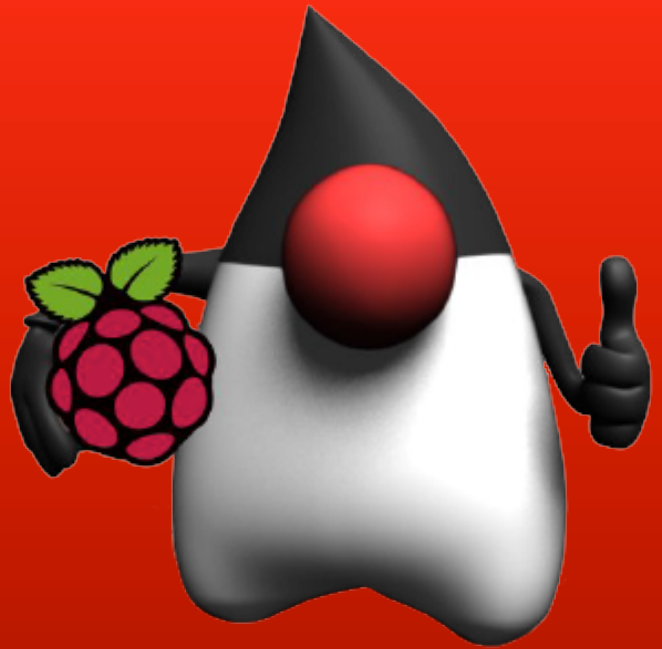


The Wiring

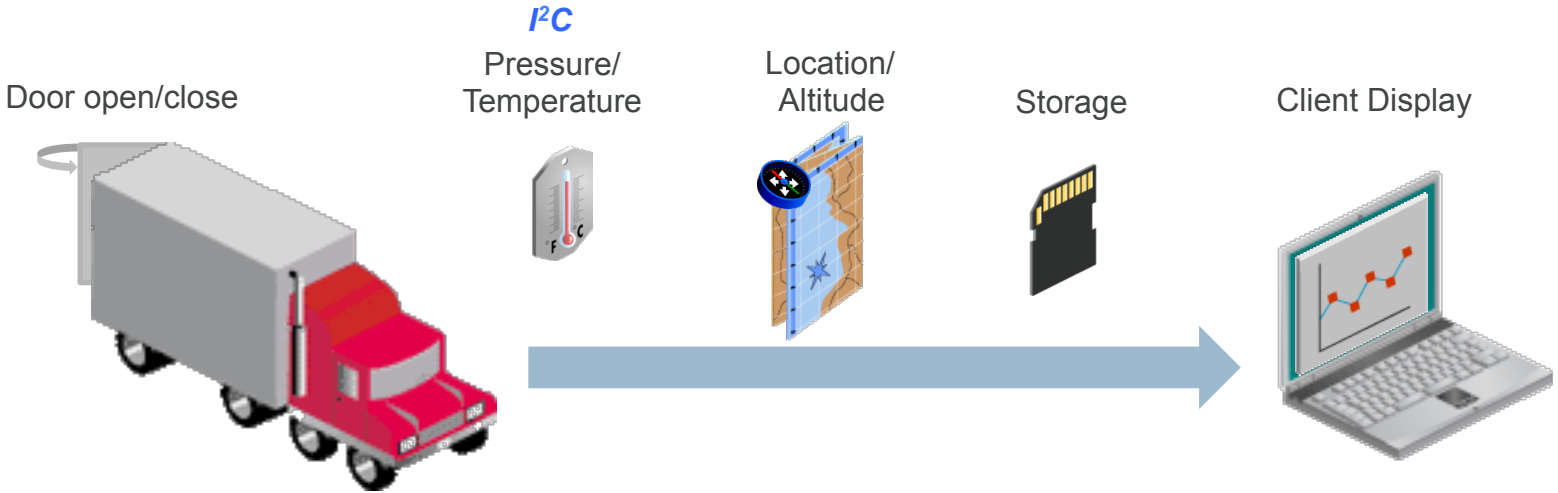


ORACLE

Lesson 2.4: I2C Overview



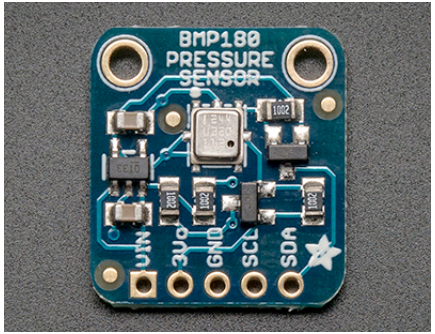
Pressure / Tempurature



In this course, you will build a prototype of an embedded device to collect, store, analyze and share data from a shipping container.

Hardware

BMP180



- Measure barometric pressure and temperature
- Specifications:
  - Pressure sensing range: 300-1100 hPa (9000m to -500m above sea level)
  - Up to 0.03hPa / 0.25m resolution
  - -40 to +85°C operational range, +-2°C temperature accuracy
  - 2-pin I^2C interface on chip
  - Uses 3.3-5V power and logic level for more flexible usage

I²C Overview

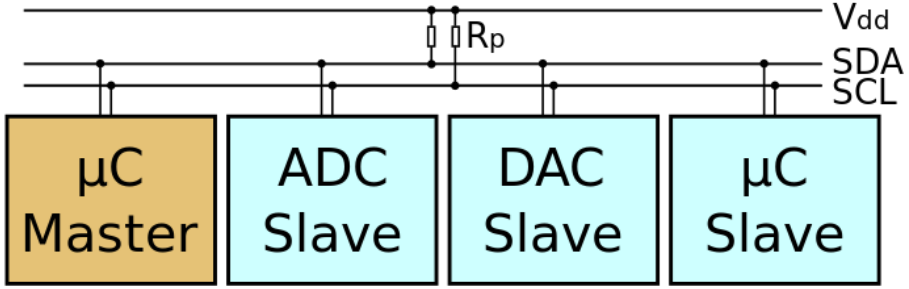
- Used for moving data simply and quickly from one device to another
- Serial Interface
- Synchronous:
  - The data (SDA) is sent along with a clock signal (SCL)
  - The clock signal controls when data is changed and when it should be read
  - Clock rate can vary, unlike asynchronous (RS-232 style) communications
- Bidirectional

I²C In Few Words…

- Inter-Integrated Circuit, ("two-wire interface")
- Multi-master serial single-ended computer bus
- Invented by Philips for attaching low-speed peripherals to a motherboard, embedded system, cellphone, or other electronic device.
- Uses two bidirectional open-drain lines
  - Serial Data Line (SDA) and
  - Serial Clock (SCL),
  - pulled up with resistors.
- Typical voltages used are +5 V or +3.3 V

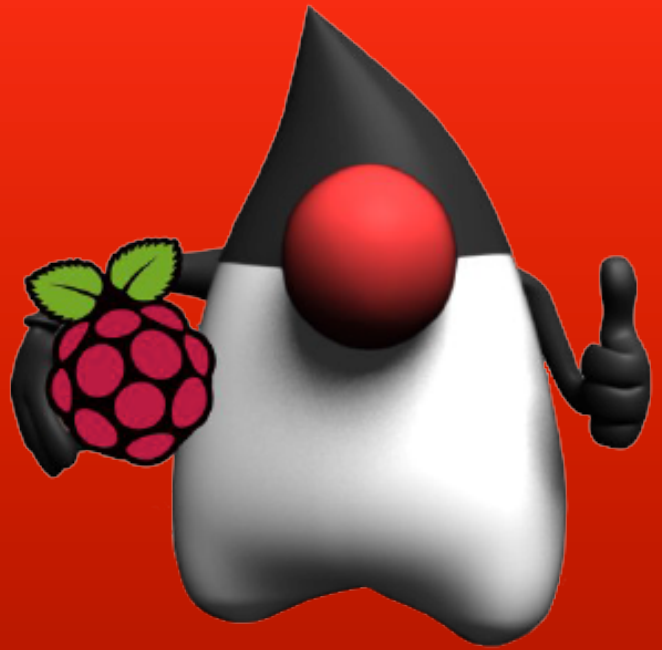
I²C Design

- A bus with a clock (SCL) and data (SDA) lines
- 7-bit addressing.
- Roles:
  - Master: generates the clock and initiates communication with slaves
  - Slave: receives the clock and responds when addressed by the master
- Multi-Master bus
- Switch between master and slave allowed



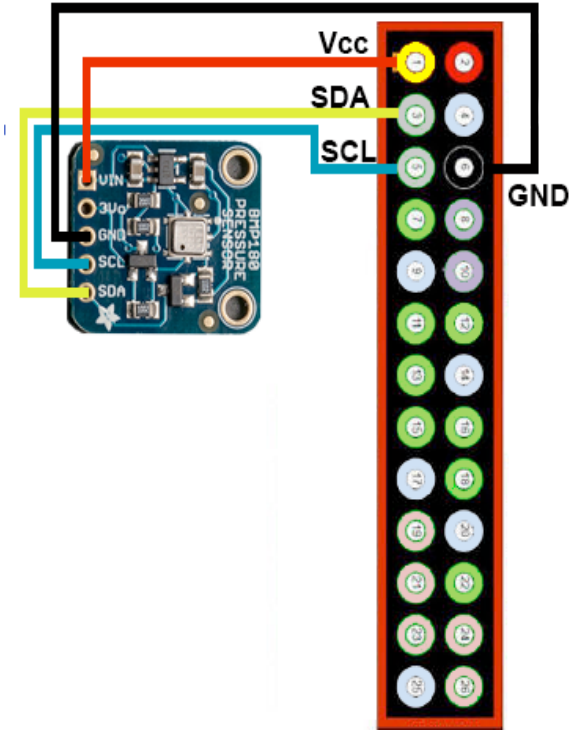
ORACLE

Lesson 2.6:
Using I²C on the Raspberry Pi

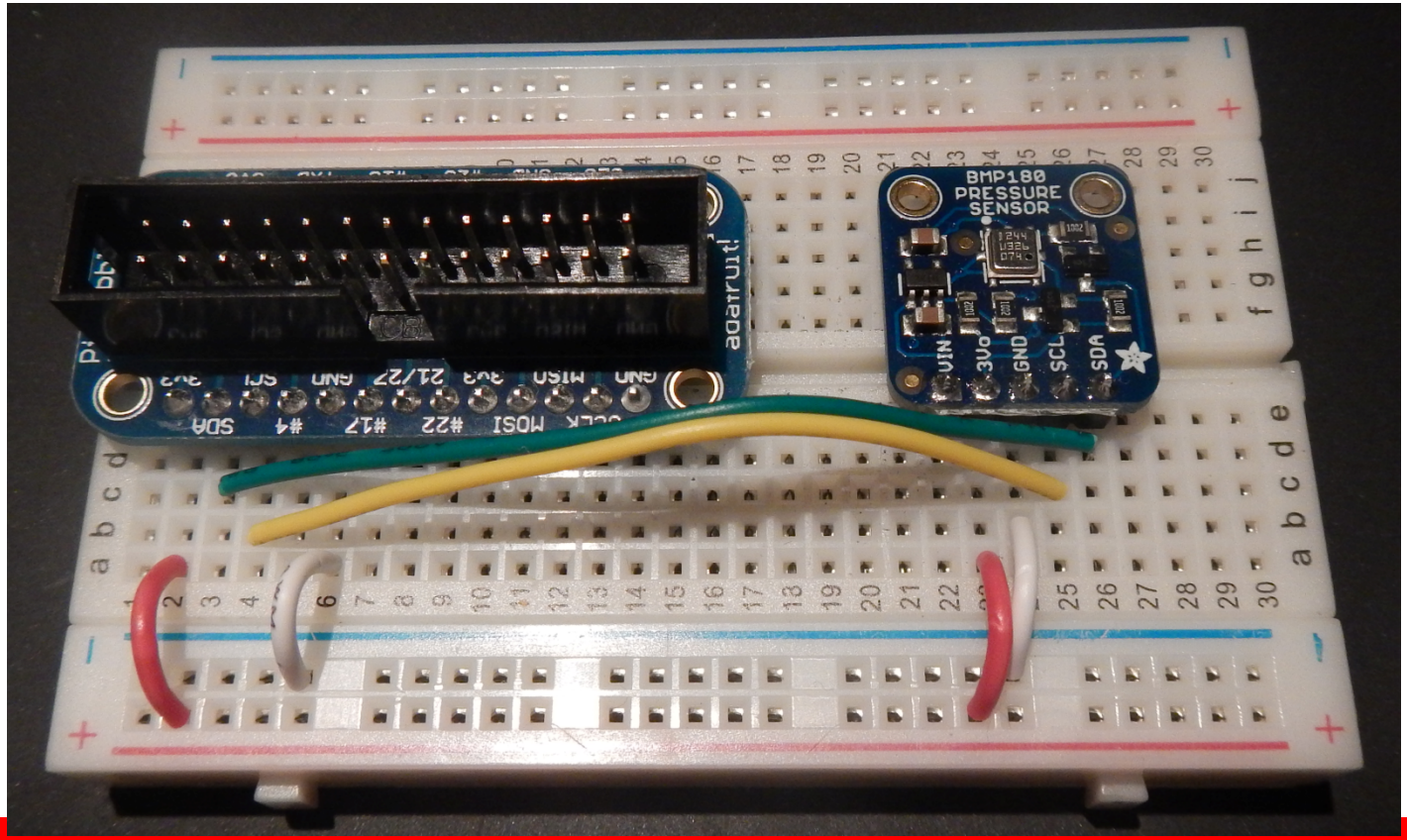


Wiring and operating the BMP180

- Raspberry Pi uses I²C bus 1
- We're using device address 0x77
- I²C uses 7 bits for device address
- BMP180 uses clock frequency 3.4MHz
- Control register for BMP180 is 0xF4
- Temperature value is at register 0xF6.



The Wiring

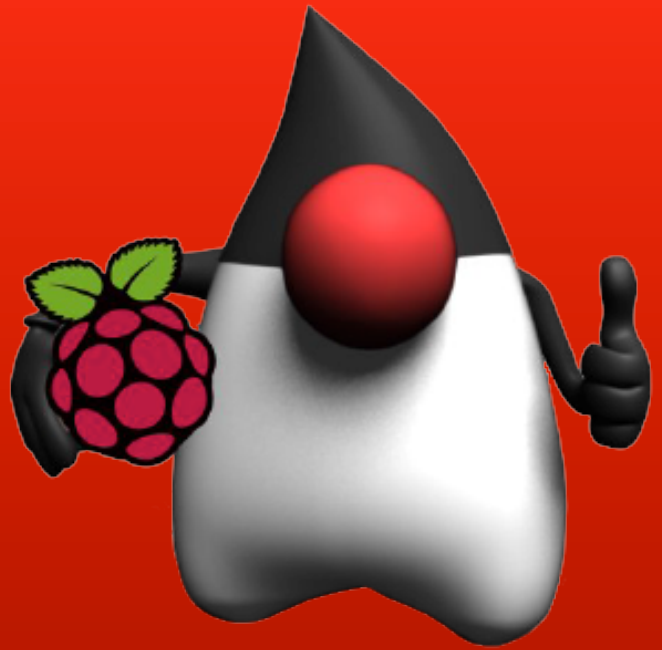


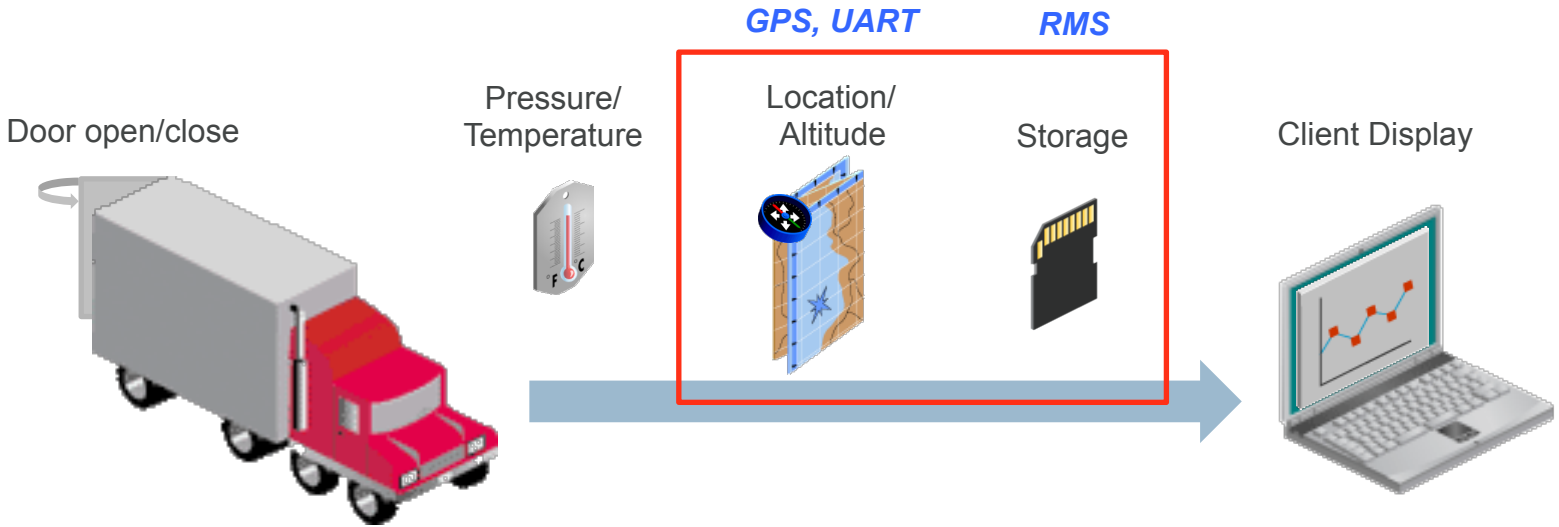
Reading and writing to I2C devices

```
I2CDeviceConfig config =
  new I2CDeviceConfig(1,              // I2C bus number
                      0x77,           // I2C device address
                      7,              // address size in bits
                      3400000);       // Clock frequency

myDevice = DeviceManager.open(config);

myDevice.write(0xF4, 1, 0x2E); // control register, size,
                               // get temperature command

myDevice.read(0xF6, 1, buf); // temperature register,
                             // subaddress size, ByteBuffer
```

ORACLE

Lesson 3: Using the UART, GPS, and Storing Data





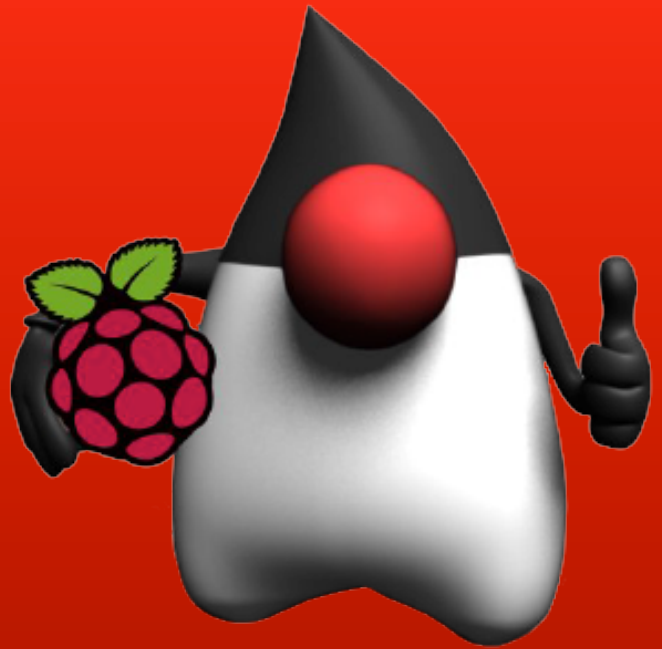
In this course, you will build a prototype of an embedded device to collect, store, analyze and share data from a shipping container.

Lesson Agenda

- UART serial protocol overview
- GPS overview
- Connecting the GPS sensor using a USB UART cable
- Connecting the GPS directly to the Raspberry Pi UART
- Reading GPS data in a Java ME 8 Embedded application
- Saving GPS data in the Record Management Store
- Saving GPS data in a file

ORACLE

Lesson 3-1:
UART Serial Protocol Overview



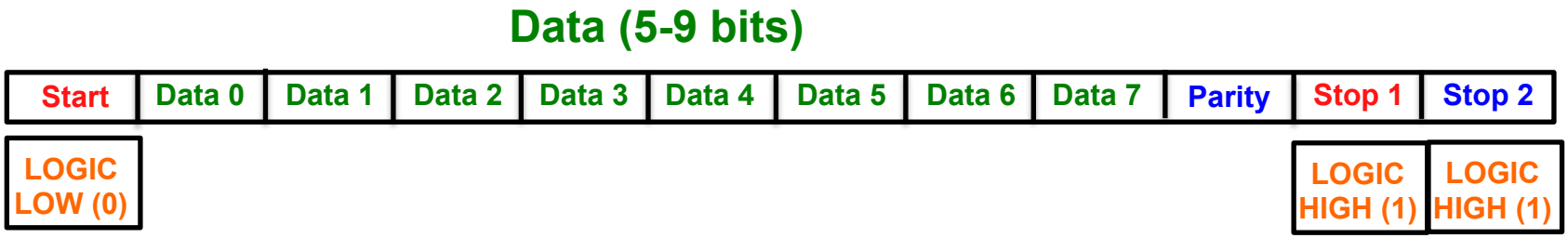
UART Serial Protocol

The Basics

- UART = Universal Asynchronous Receiver and Transmitter
- Serial communication
  - Each bit is sent one after the other
- Both ends must be set to work at the same speed
  - bits per second
  - Standard speeds (9600, 19200, 115200, etc)
  - Some devices use non-standard speeds
- At idle, with no data the connection is held high (voltage)

UART Protocol

Character Frame

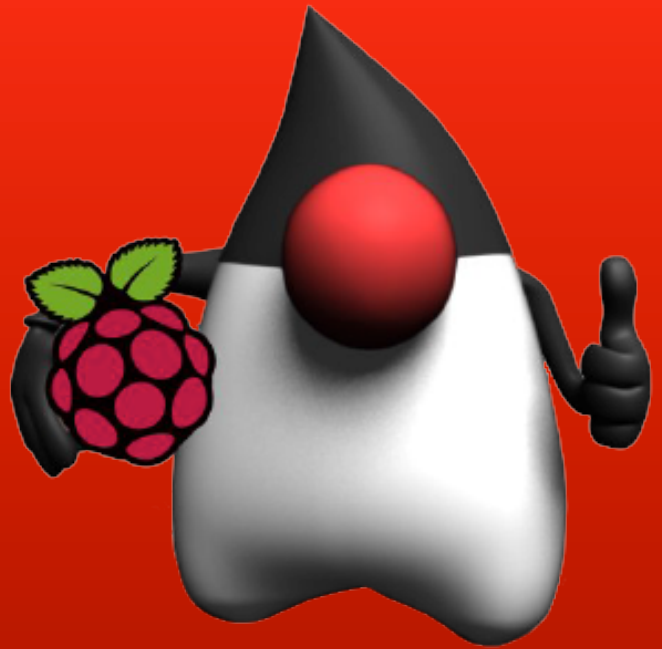


Serial Data Transmission

- UART does not usually drive transmission of data
  - RS-232, RS-422, RS-485
- Transmission does not need to be electrical
  - Bluetooth serial port profile
  - Infra red (IRDa)
  - Acoustic modem (telephone line)

ORACLE

Lesson 3-2: GPS Overview



Global Positioning System

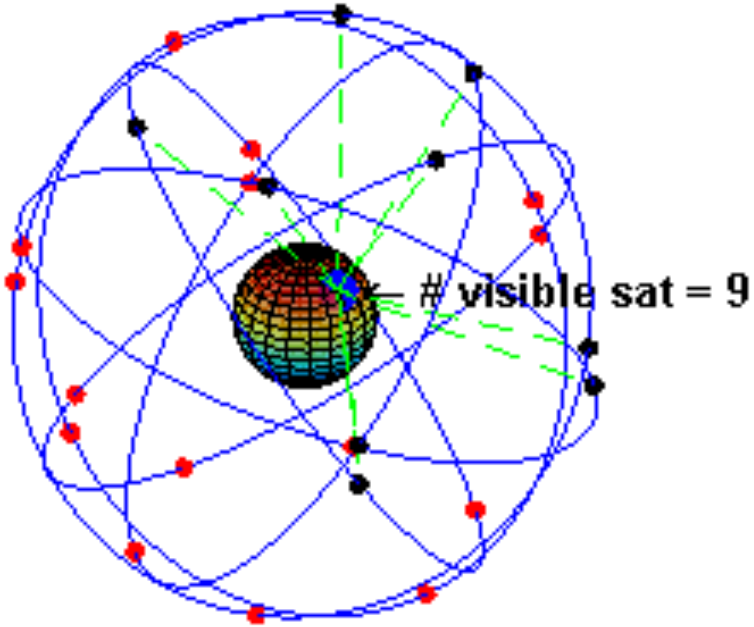


Image courtesy of Wikipedia

GPS

Satellite Navigation System

- Position is calculated using satellites that continually transmit data
  - Time and the position above the earth
- Minimum of three satellites required to get position
  - latitude and longitude
- Four satellites required to get 3D position fix
  - latitude/longitude/altitude

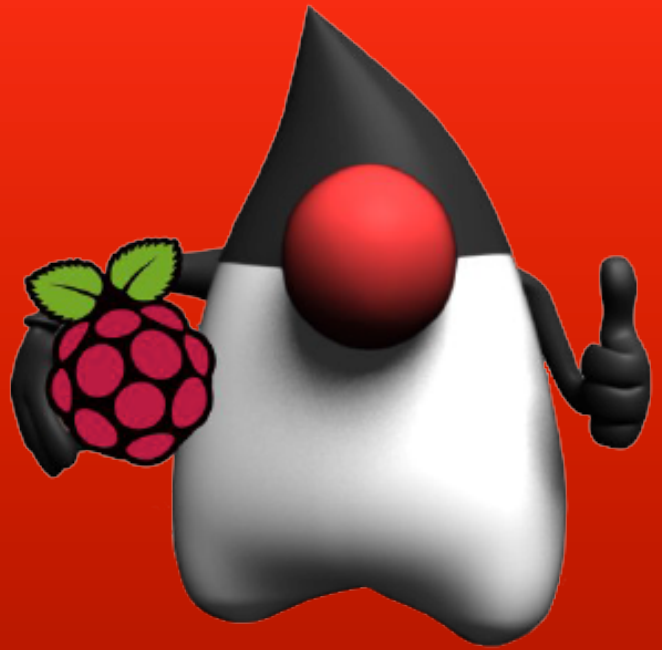
GPS Receiver

Data Format

- GPS receivers typically use NMEA 0183 standard
  - National Marine Electronics Association
- Text based protocol
  - Message starts with a $ sign
  - Next five letters identify the talker (2 letters) and type (3 letters)
    - GPS receiver talker code is GP
  - Data fields are comma separated
  - Message contains optional aterisk * and two digit checksum at the end
  - Message is terminated with carriage return, line feed (13, 10 ASCII)

ORACLE

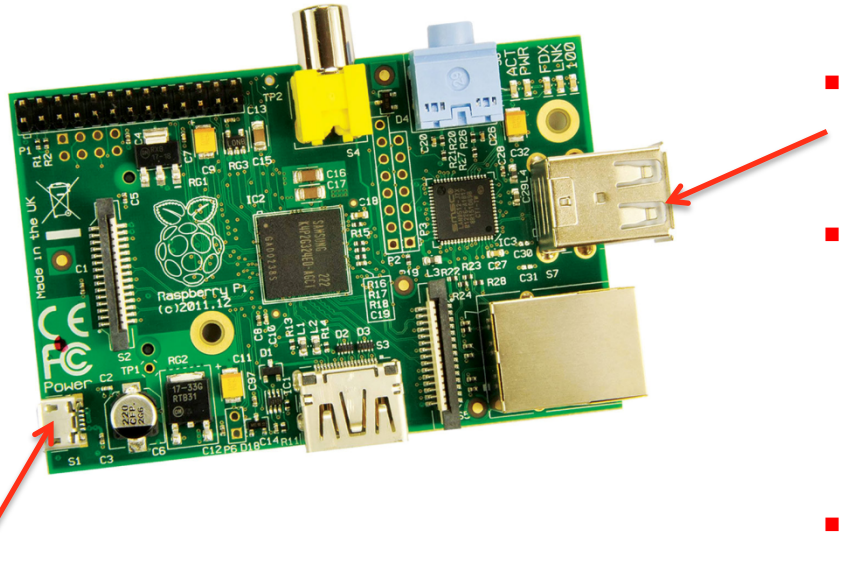
Lesson 3-3: Connecting the GPS Sensor To The Raspberry Pi



Connection Methods

1. Using a USB to TTL UART adapter
2. Direct connection to the Raspberry Pi header pins

Raspberry Pi USB Connectors

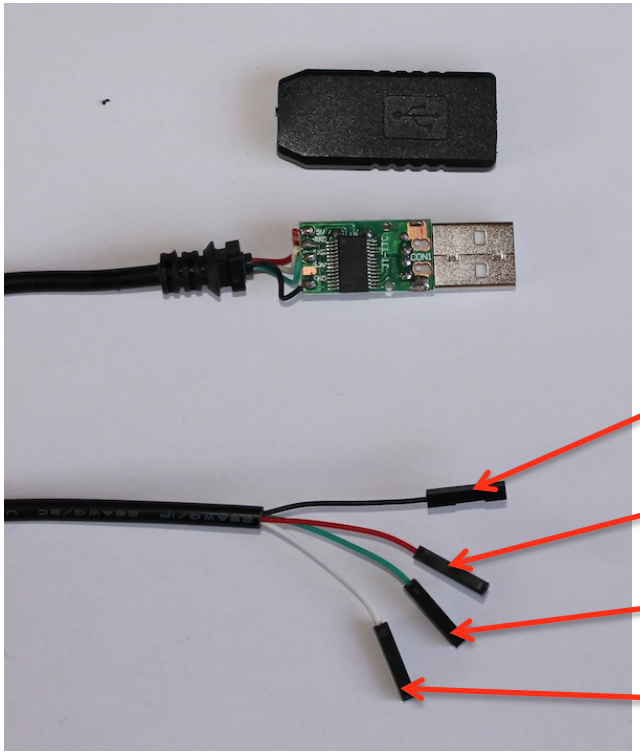


- 2 x USB 2.0 Type A sockets
  - Host USB
- Differs slightly from USB specification
  - Power should be 500mA max
  - Actual is less than 200mA
- Can power the board through these

- Micro USB socket
  - Power only

USB UART Cable

PL2003 UART TTL-USB



USB Adapter
(Vcc can be +3.3V or +5V)

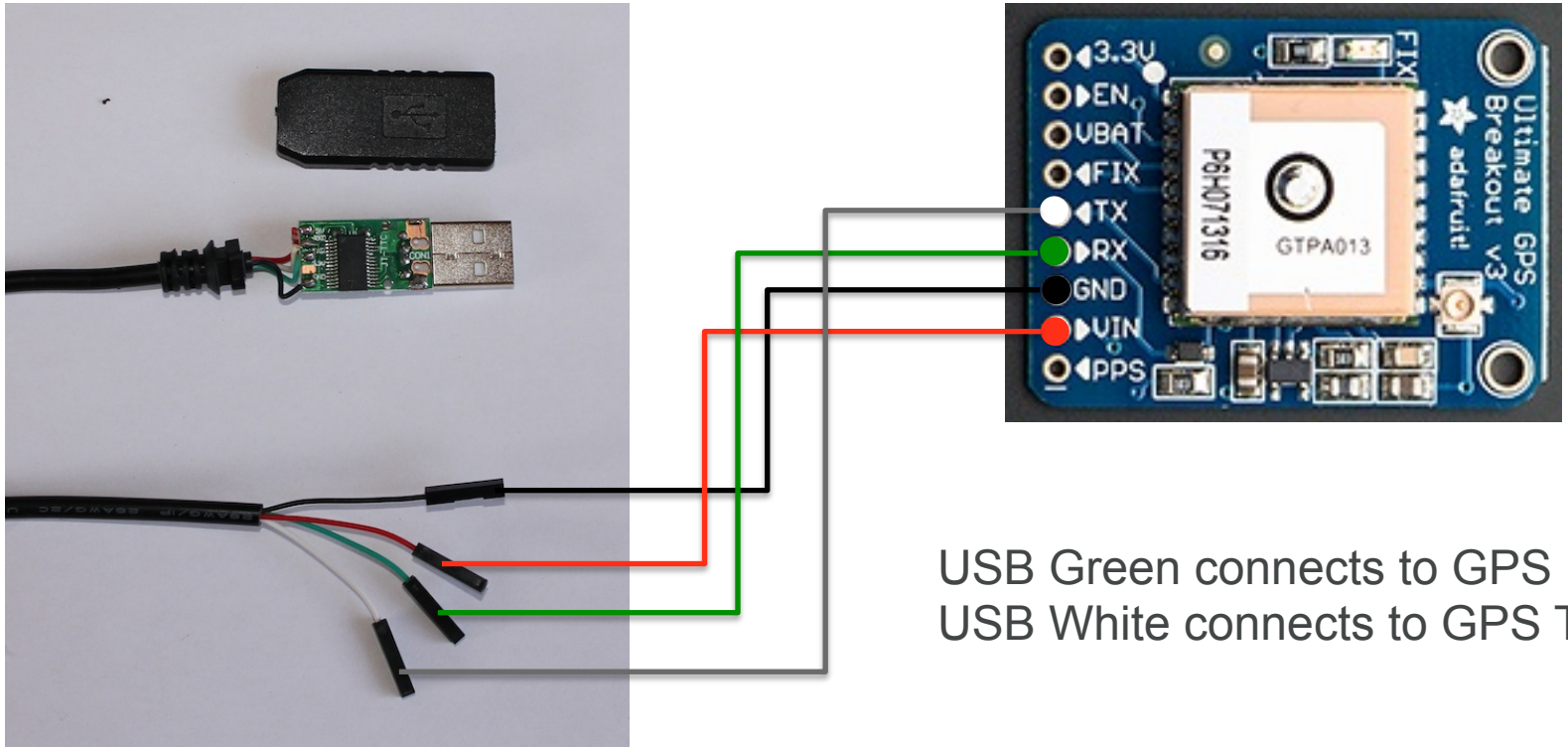
GND (Black)

Vcc +5V (Red)

TX (Green)

RX (White)

USB UART Adapter Connections



USB Green connects to GPS RX
USB White connects to GPS TX

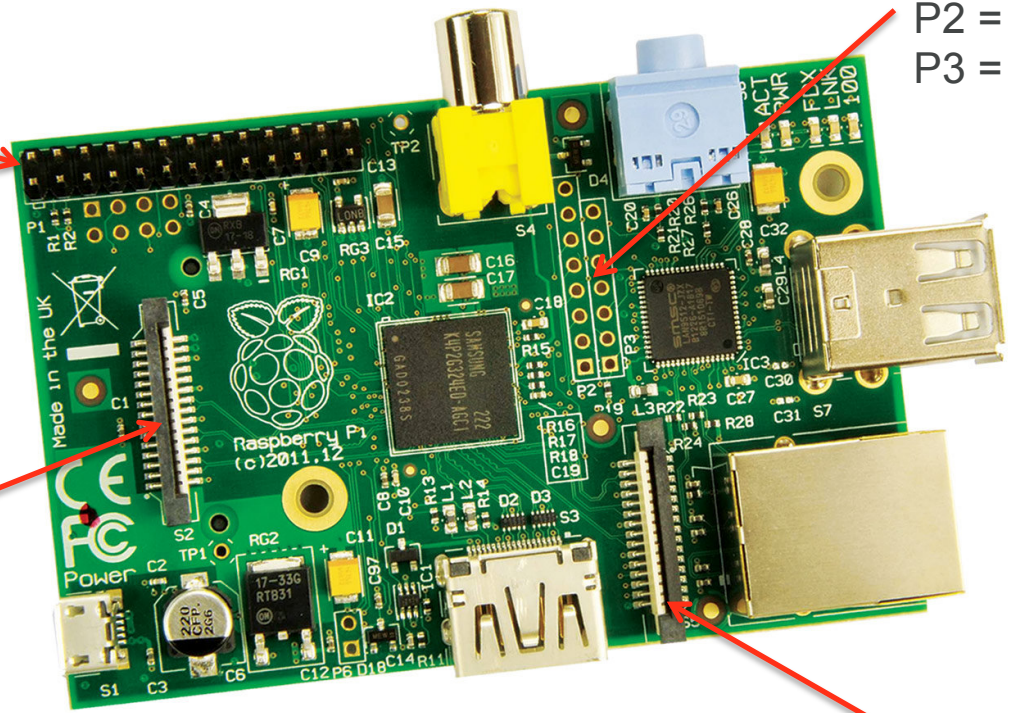
USB UART Raspberry Pi Configuration

- Plug in USB UART adapter to Raspberry Pi
- Device to use is **`/dev/ttyUSB0`**

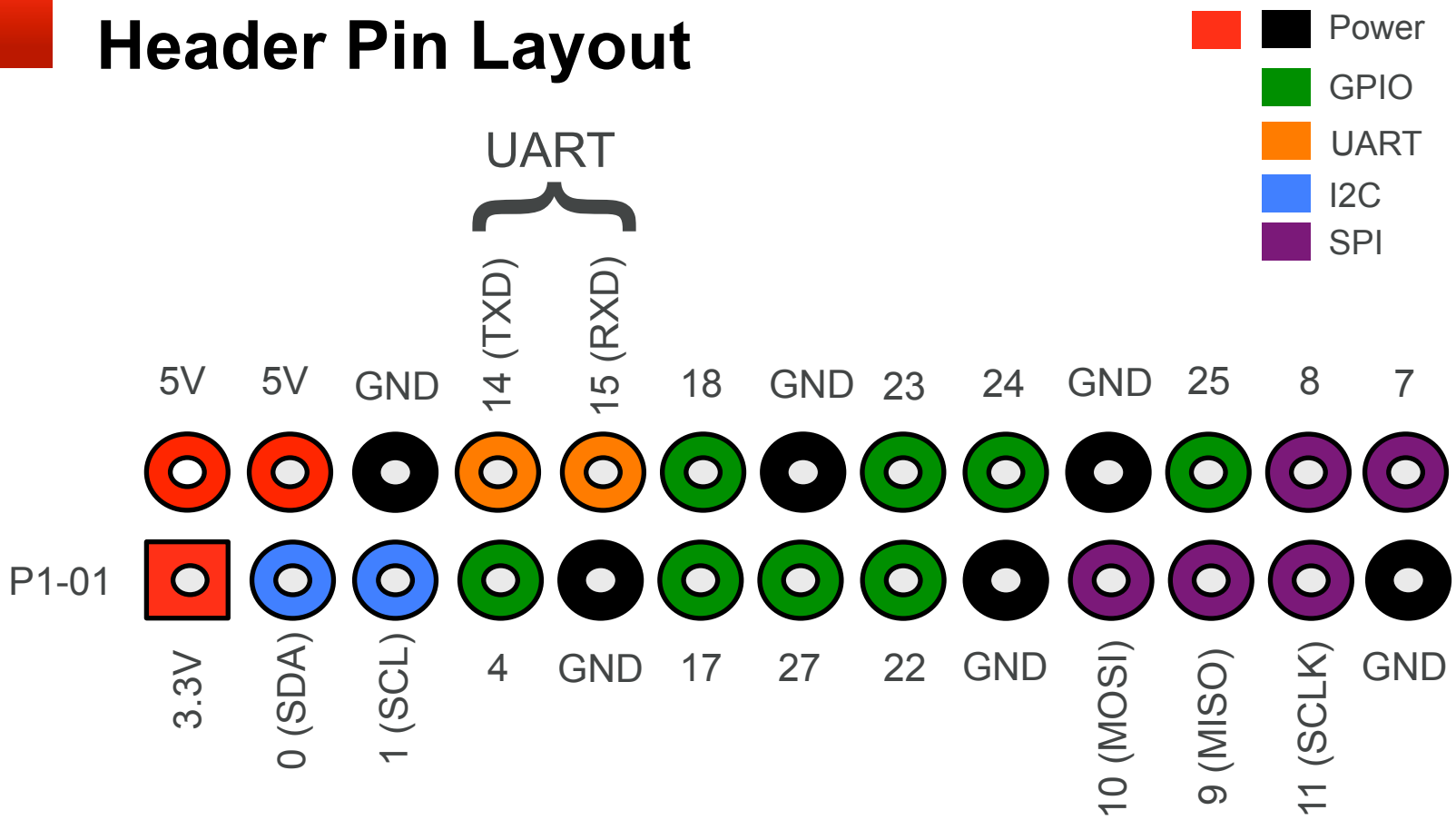
Raspberry Pi Headers

Connections (P1)
- GPIO
- I2C
- SPI
- UART

Display (Not supported)

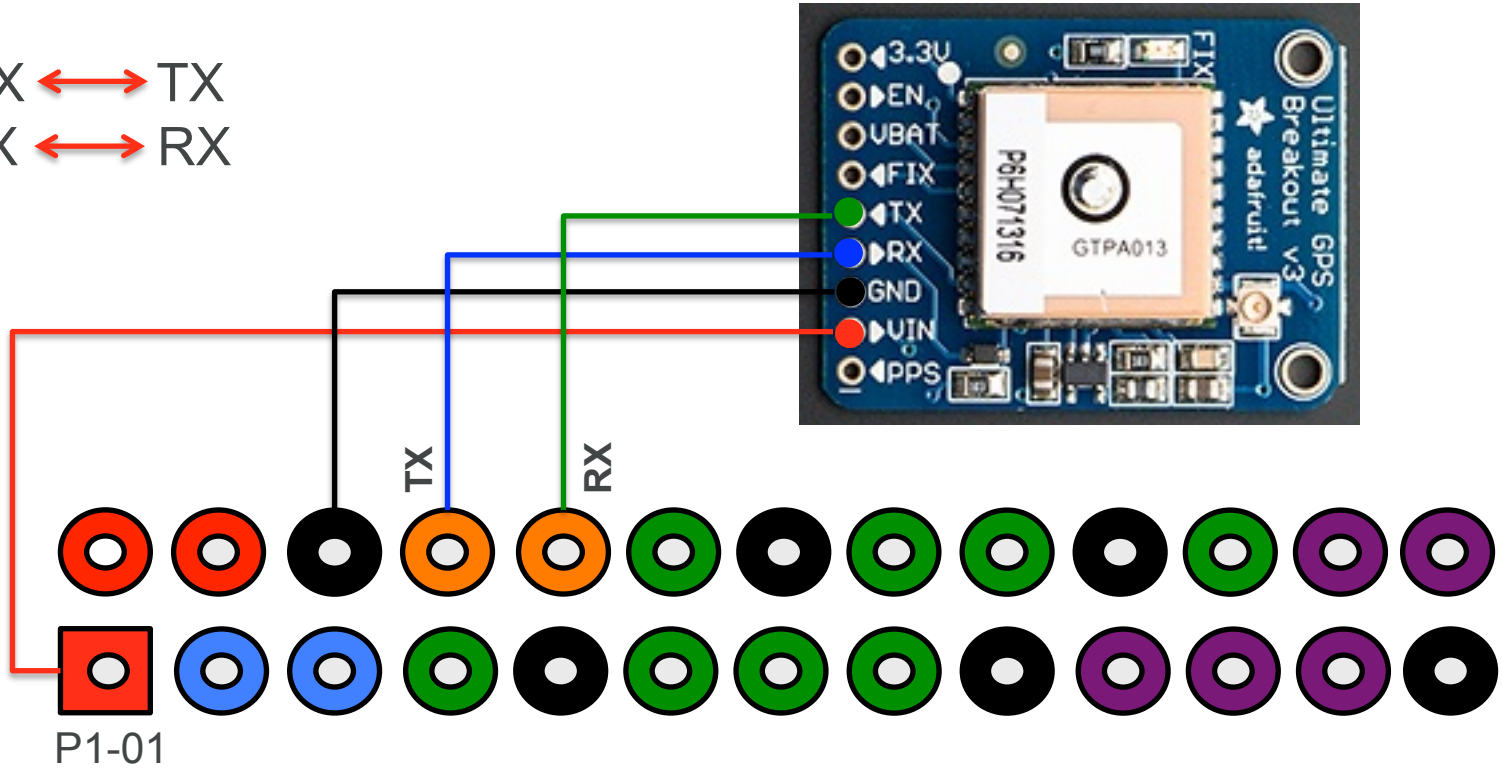


Header Pin Layout



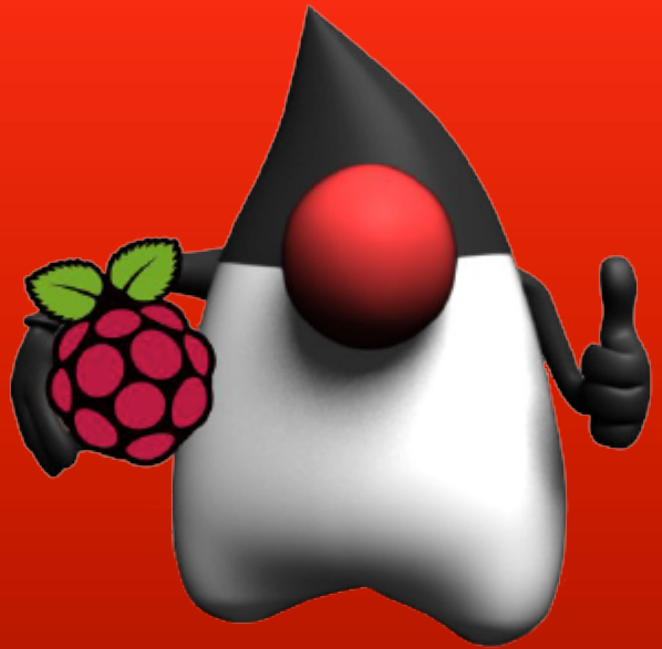
UART Connections

NOTE: RX ⟷ TX
TX ⟷ RX



ORACLE

Lesson 3-4: Reading GPS Data in a Java ME 8 Embedded Application



Using The UART

Getting a BufferedReader

```
UART uart = DeviceManager.open(40); //UART ID #
uart.setBaudRate(9600);
InputStream is = Channels.newInputStream(uart);
InputStreamreader isr = new InputStreamReader(is);
BufferedReader reader = new BufferedReader(isr);
```

UART Device Security

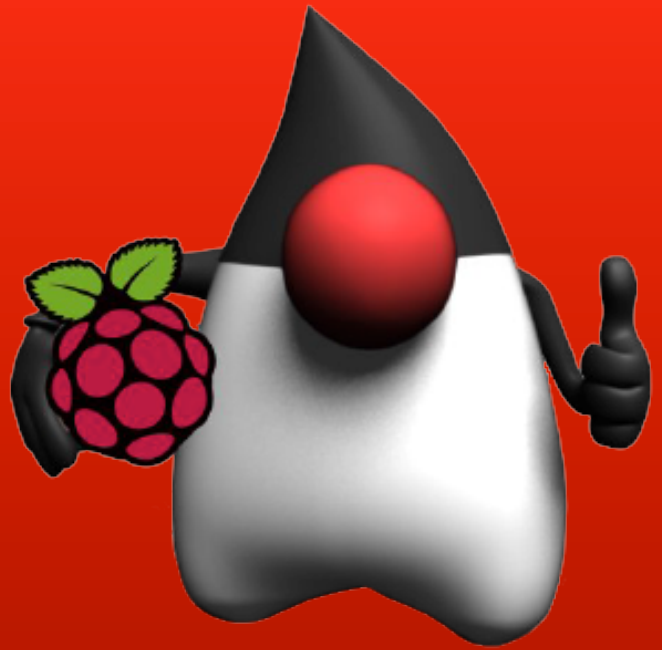
- Need to add API permission
  - Application Descriptor
  - Permission
    - `jdk.dio.DeviceMgmtPermission`
  - Protected resource name
    - `*:*`
  - Actions requested
    - `open`

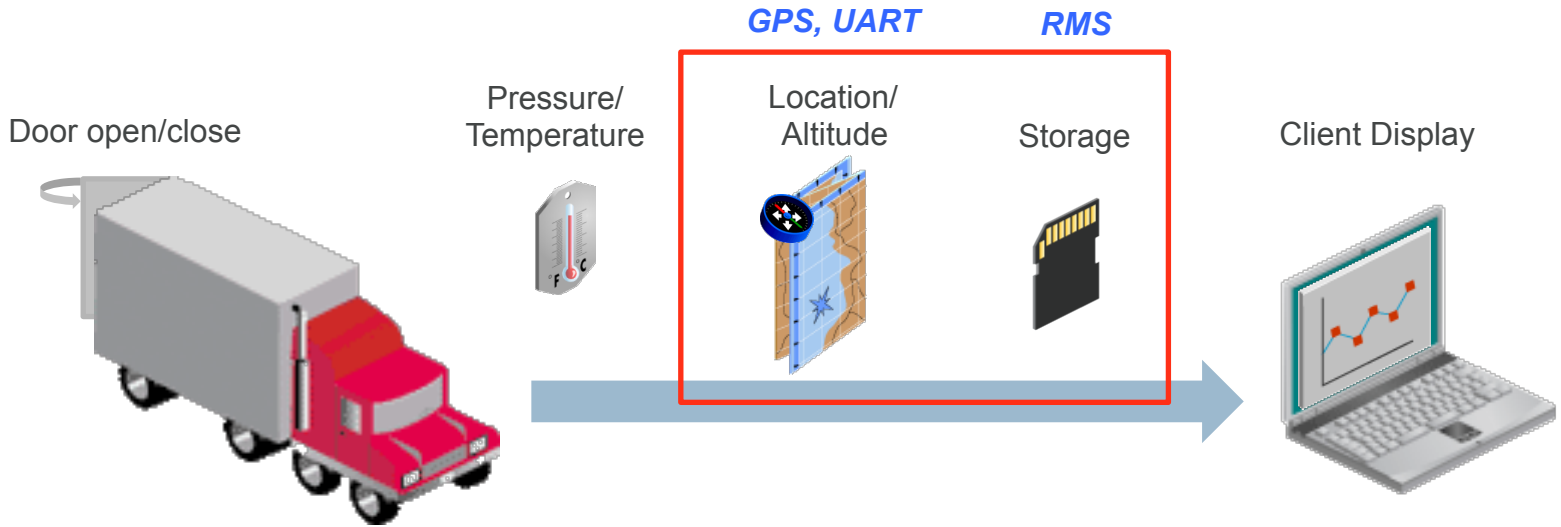
Reading GPS Data

- All data lines start with $
  - Read lines until we get one with the right start character
- Check the talker is GP
- Test for the correct type (GGA or VTG)
- Decode comma separated values
- Data appears to be easily corrupted
  - Short reads, $ in middle of line, not enough fields
  - Repeat reads until valid data extracted

ORACLE

Lesson 3-5: Saving GPS Data in the Record Management System

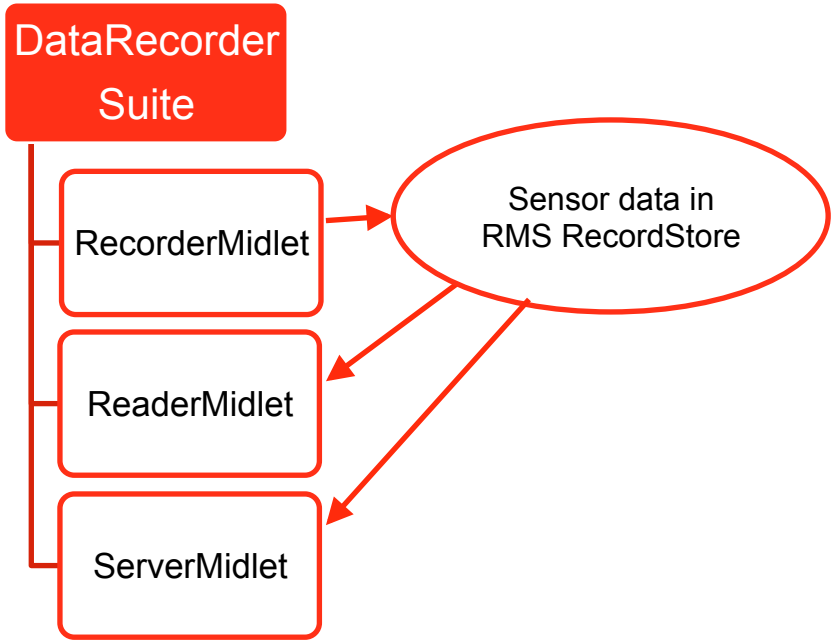




In this course, you will build a prototype of an embedded device to collect, store, analyze and share data from a shipping container.

Our Midlet suite

Stores and retrieves sensor data



Record Management System (RMS) Overview

Defined in Java ME Embedded Profile 8.0

- Java ME does not rely on being able to store data in files
  - Some small and embedded devices do not have file systems
- Java ME stores application data in non-volatile memory
  - How this happens is abstracted away by the RMS
- RMS can be thought of as a very simple database
  - Two columns
    - Record ID
    - Data (array of bytes)

RecordStore

Storing Data

```
RecordStore store = RecordStore.openRecordStore("gps-data", true);

String data = position.toString() + "^" + velocity.toString();
byte[] dataBytes = data.getBytes();

int recordNum = store.addRecord(dataBytes, 0, dataBytes.length);
```

RecordStore

Retrieving Data

- Create a new Midlet in the same Midlet suite as the data recorder
- Records are retrieved as **`byte`** arrays

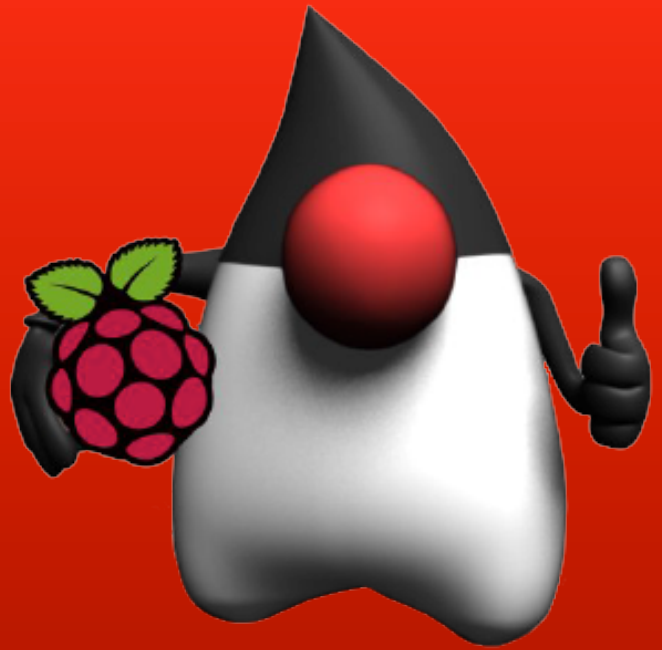
```
RecordStore store = RecordStore.openRecordStore("gps-data", false);
recordCount = store.getNumRecords();

for (int i = 1; i < recordCount; i++) // record index is 1 based
  System.out.println("Record " + i + " = " +
    new String(store.getRecord(i));
```

Record Store Persistence

- The persistence of the Record Store is tied to the MIDlet suite
- When the MIDlet suite is removed from the device (destroyed)
  - The Record Store for that suite is deleted

ORACLE

Lesson 3-6:
Saving GPS Data in a File



Java ME And Files Overview

- Raspberry Pi has conventional filesystem
  - So data can be stored there
  - Java ME supports this through the Generic Connection Framework
- Be careful with file URI
  - Format is **`file://[root]/[file-path]`**
  - Example file, **`/tmp/gps-data.txt`**
  - URI is **`file://rootfs/tmp/gps-data.txt`**

Java ME File IO

Generic Connection Framework: FileConnection

```
private final FileConnection connection;
private final PrintStream fileWriter;

connection =
  (FileConnection)Connector.open(
     "file://rootfs/tmp/gps-data.txt",
     Connector.READ_WRITE);

if (!connection.exists())
  connection.create();
```

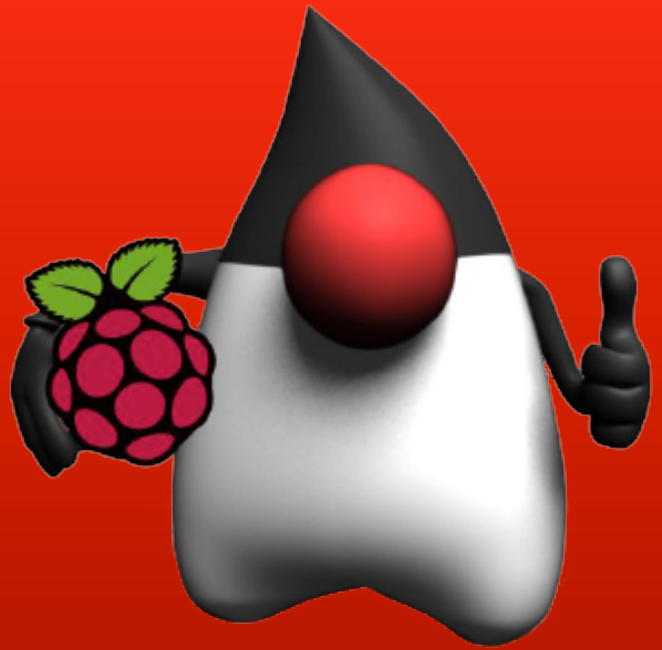
Java ME File IO

Use Generic Connection Framework

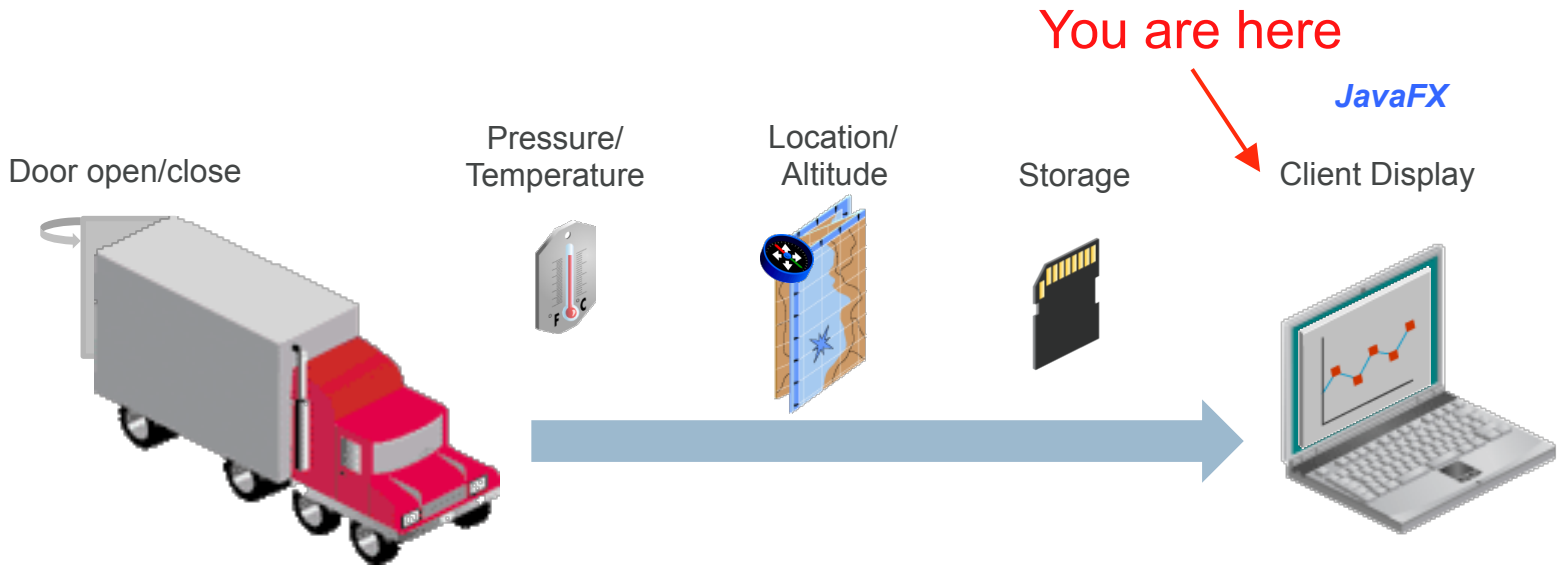
```
fileWriter = new PrintStream(
  connection.openOutputStream());

fileWriter.println(position + "^" + velocity);
```

ORACLE

Lesson 4: Communicating the Data to the Outside World



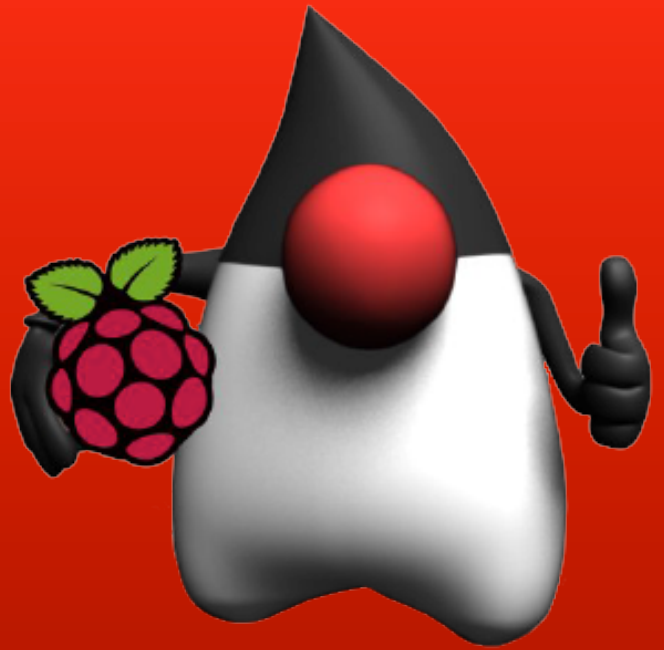
In the grand scheme of this course …



In this course, you are building a prototype of an embedded device to collect, store, analyze and share data from a shipping container.

ORACLE

Lesson 4-1:
Demo a MIDlet that serves sensor data, and a client app that communicates with it



Two more pieces of the prototype puzzle …

An app named VisualClient that communicates with ServerMidlet

DataRecorder Suite

RecorderMidlet

ReaderMidlet

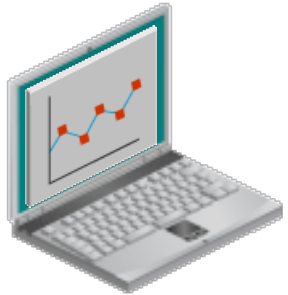
ServerMidlet

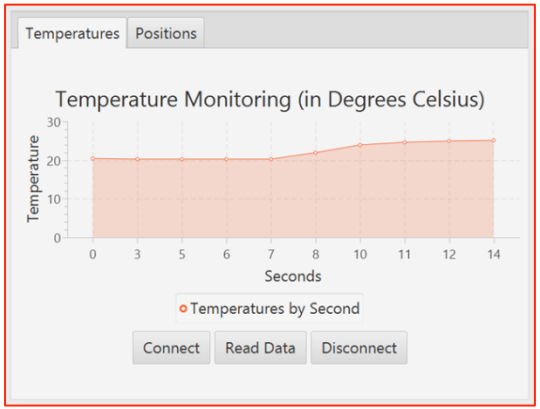
Sensor data in RMS RecordStore

"CONNECT", "READ", "DISCONNECT"

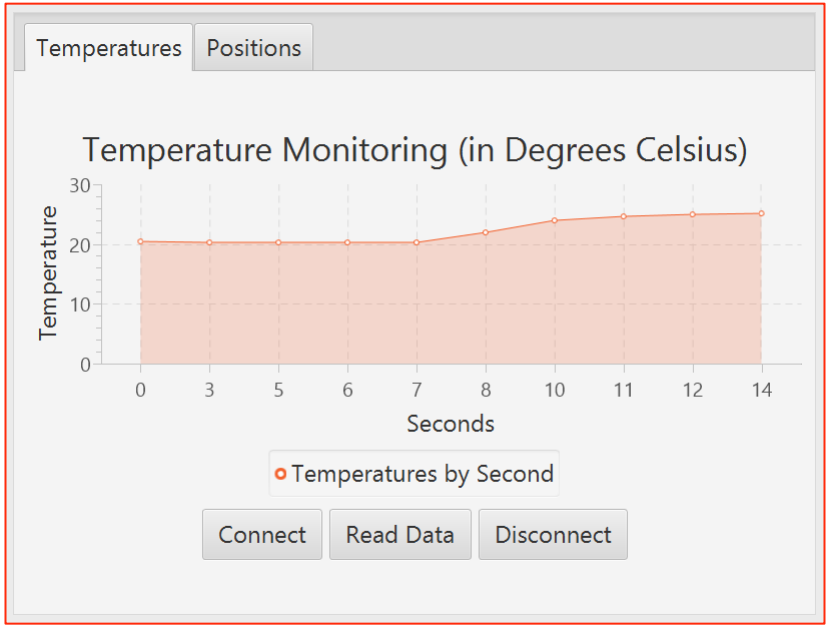
Port 8042

CSV data stream

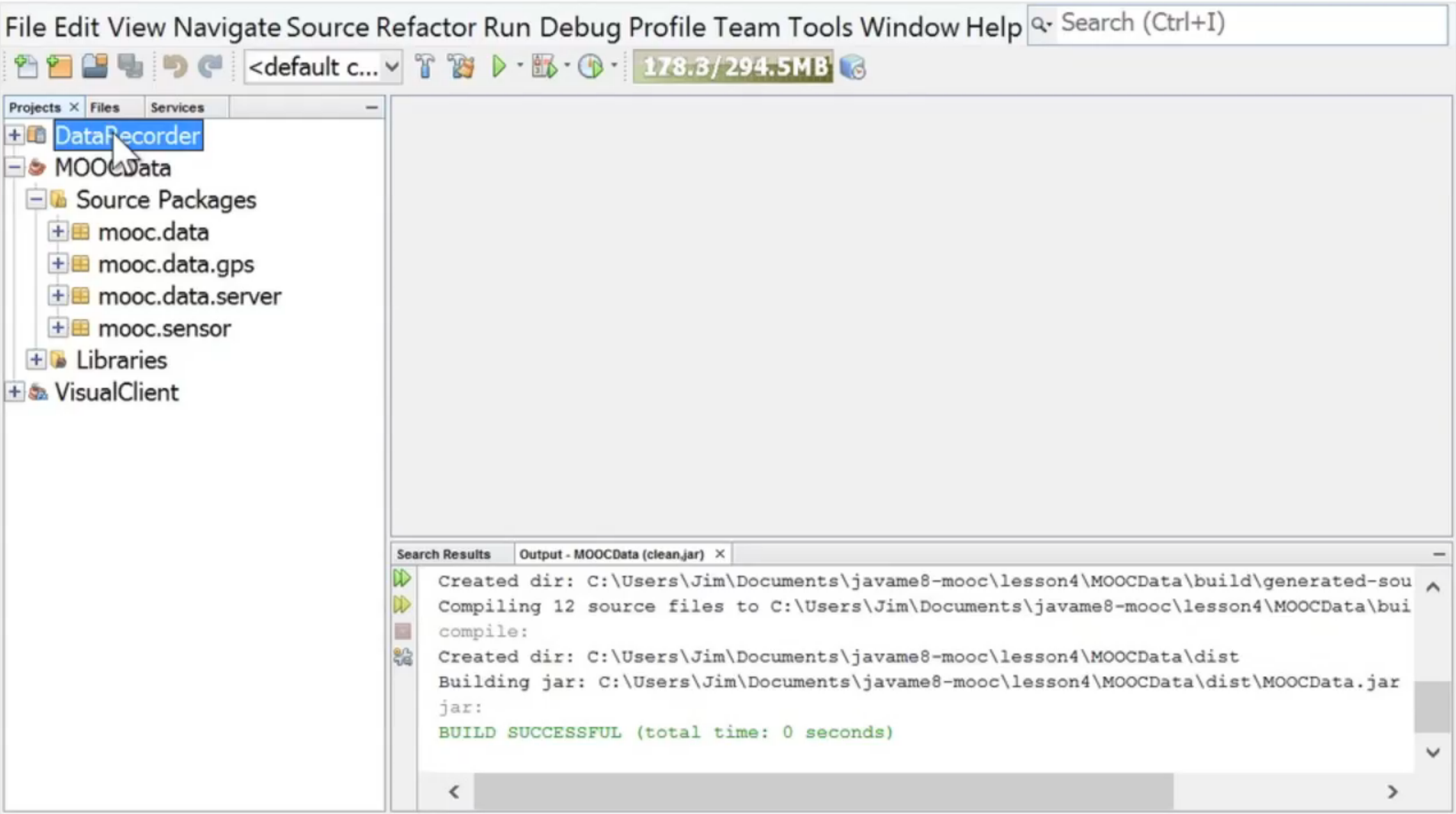




Taking ServerMidlet and VisualClient for a spin



VisualClient is created with Java/JavaFX and the SceneBuilder tool


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default c...
178.3/294.5MB
Projects
Files
Services
DataRecorder
MOOCData
Source Packages
mooc.data
mooc.data.gps
mooc.data.server
mooc.sensor
Libraries
VisualClient
Search Results
Output - MOOCData (clean,jar)
Created dir: C:\Users\Jim\Documents\javame8-mooc\lesson4\MOOCData\build\generated-sou
Compiling 12 source files to C:\Users\Jim\Documents\javame8-mooc\lesson4\MOOCData\bui
compile:
Created dir: C:\Users\Jim\Documents\javame8-mooc\lesson4\MOOCData\dist
Building jar: C:\Users\Jim\Documents\javame8-mooc\lesson4\MOOCData\dist\MOOCData.jar
jar:
BUILD SUCCESSFUL (total time: 0 seconds)

Raspberry Pi with Java

a) **Intro to Raspberry Pi and Java ME 8**
b) **Raspberry Pi Tricks**

#RPiJava8
#DevNexus

James Weaver @JavaFXpert
Stephen Chin @SteveOnJava
Java/IoT/Cloud Technology Ambassadors
Oracle Corporation

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.