CPEN 291

Computer Engineering Design Studio I

Lab 4:

Arduino Weather Monitoring System

(Interfacing Arduino with Other Software: MATLAB, Processing)

And Introduction to Raspberry Pi 3

ECE – UBC

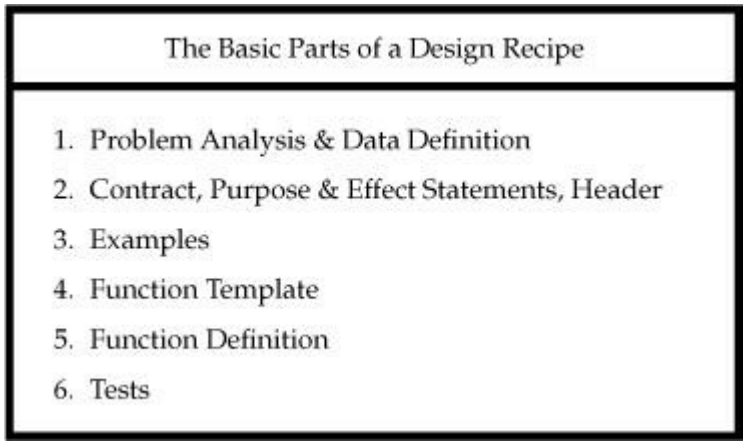
2016 W2

January-February 2017

Electrical and Computer Engineering
University of British Columbia

Notes on Program Design

- At a simple form, the basic steps/parts of a program design recipe are:



- Each step produces a well-defined intermediate product:

1. the description of the class of problem data;
2. the informal specification of a program's behavior;
3. the illustration of the behavior with examples;
4. the development of a program template or layout;
5. the transformation of the template into a complete definition; and
6. the discovery of errors through testing.

Source: *How to Design Programs*, MIT Press

Objective

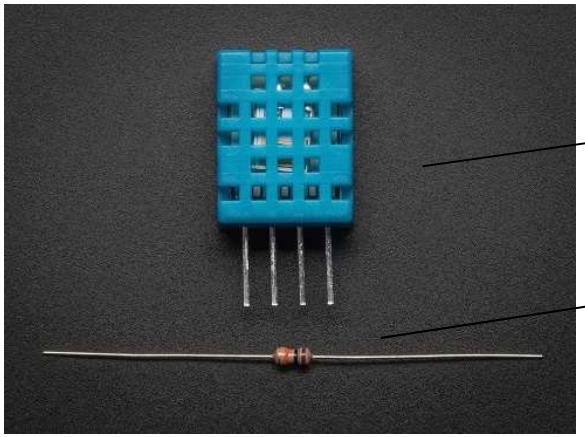
- The objective of this lab is
  - to interface Arduino with other software to graph or further process the obtained sensor data, and
  - to implement a simple weather monitoring system (humidity, temperature …) as an application.

- We will use the following two software:
  - Processing
  - MATLAB (alternatives are *Octave* and *Scilab*)

- Interfacing: The digital humidity sensor itself communicates with Arduino via a single IO pin.
- We will also have an introduction to Raspberry Pi 3.

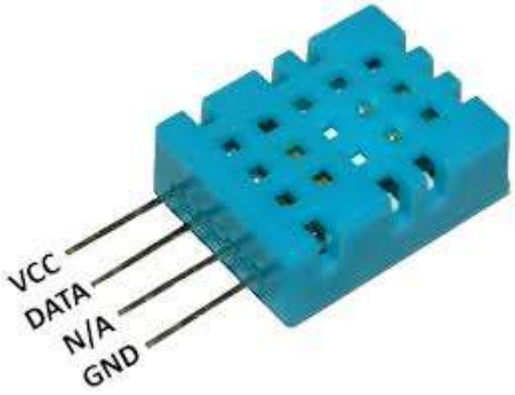
Digital Humidity Sensor

- DHT11 is a digital humidity/temperature sensor that is used in this lab. It is a basic, inexpensive and rather slow sensor. The sensor package itself includes:
  - Capacitive humidity sensor
  - Thermistor
  - Circuitry to convert analog sensor data to digital and to send the digital data serially on one pin (DATA).



DHT11 info

- DHT11:
  - 20-80% humidity reading with 5% accuracy
  - 0-50^{o} Celsius temperature reading ±2^{o} C
  - 1 Hz sampling rate (once every second)
- The sensor has 4 pins, but one of them is not used (Not Connected).
- Pins: VCC, DATA and GND
  - Note that the DATA pin is connected to a single digital IO pin on the Arduino.

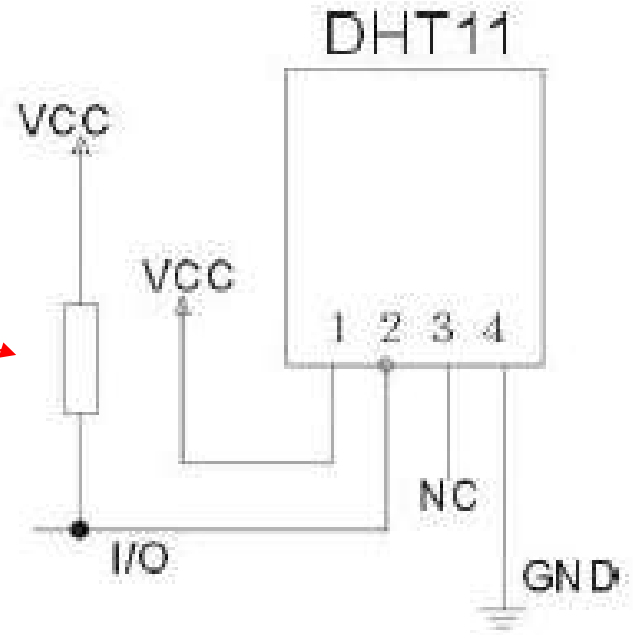


DHT11 Connection

- DHT11 uses a simplified single-pin master-slave serial communication using one pin (DATA).
- A pull-up resistor must pull the DATA pin to VCC as shown in the schematic below:

Pull-up resistor

DATA pin to an Arduino digital IO pin

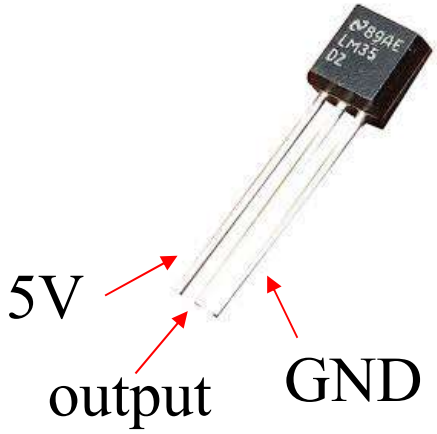


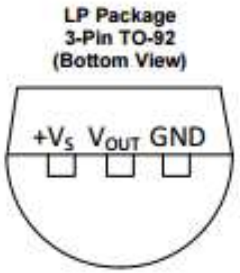
DHT11 Programming

- See the link in the lab info document for the library and example Arduino file.
  - The datasheet is poorly written/translated. Therefore, you may use the library, instead of own implementation.
- Optional: Bonus mark may be given if you fully write the DHT11 code yourself based on the datasheet
  - No copy/paste from the library code
  - Still capped to 100% mark (the bonus is used to compensate for any mark deductions somewhere else)

Analog Temperature Sensor

- LM35 linear analog temperature sensor is also used in this lab to sense the ambient air temperature.





Bottom view

- Three pins: analog output, and power (GND, 5V)
- Functional range: 0 to 100 degree Celsius
- Sensitivity about 10mV per degree Celsius
- Output voltage is linearly proportional to the temperature (Centigrade).

Arduino Analog Input Pins

- 6 channel analog-to-digital (A/D) converter
- 10 bit resolution (analogRead returning integers from 0 to 1023)
- Pins A0 to A5 for analog input
  - They can also be used as general purpose IO (GPIO).

- See details and caveats in the documentation. Examples:
  - "The Atmega datasheet also cautions against switching analog pins in close temporal proximity to making A/D readings (analogRead) on other analog pins. This can cause electrical noise and introduce jitter in the analog system. It may be desirable, after manipulating analog pins (in digital mode), to add a short delay before using analogRead() to read other analog pins."
  - "The analogRead command will not work correctly if a pin has been previously set to an output, … Similarly if the pin has been set to HIGH as an output, the pullup resistor will be set …"

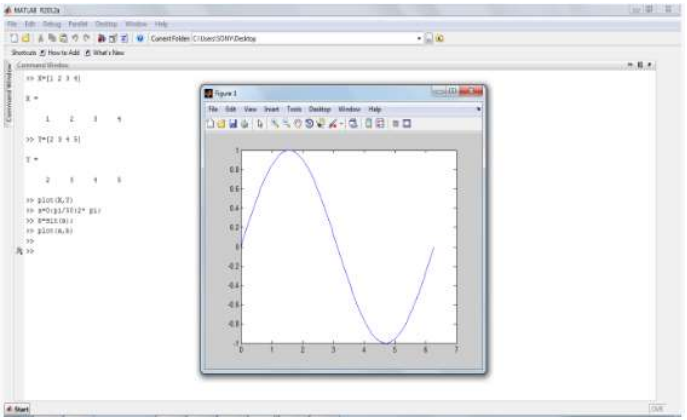
Other Components

- An on-off switch controls in what format the data is sent from Arduino to the serial monitor tool.
  - On-Off Switch:

- Photocell: You are already familiar with the photocell (Lab 1) …

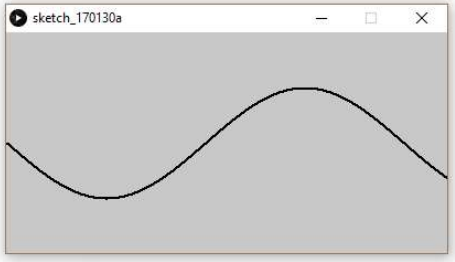
Use of other Software for Graphing

- The switch is used for the Arduino to send the sensor data with a specific format for each of the switch states.
- In one switch state, the data is also read by other software for further processing and plotting.
  - MATLAB and Processing are used for this purpose in this lab.



Sample MATLAB example

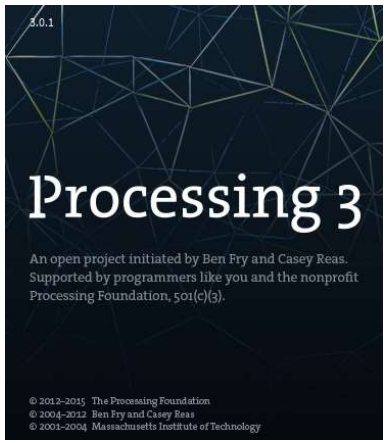
```
float angle = 0.0;
float offset = 100.0;
float scaleValue = 50.0;
float angleIncre = PI/180;

size(400, 200);
frameRate(60);
background(200);
 for (float x = 0; x <=width; x+=1) {
 float y = offset + (sin(angle) * scaleValue);
 rect(x, y, 1, 1);
 angle += angleIncre;
 }
```



Sample Processing example

Processing

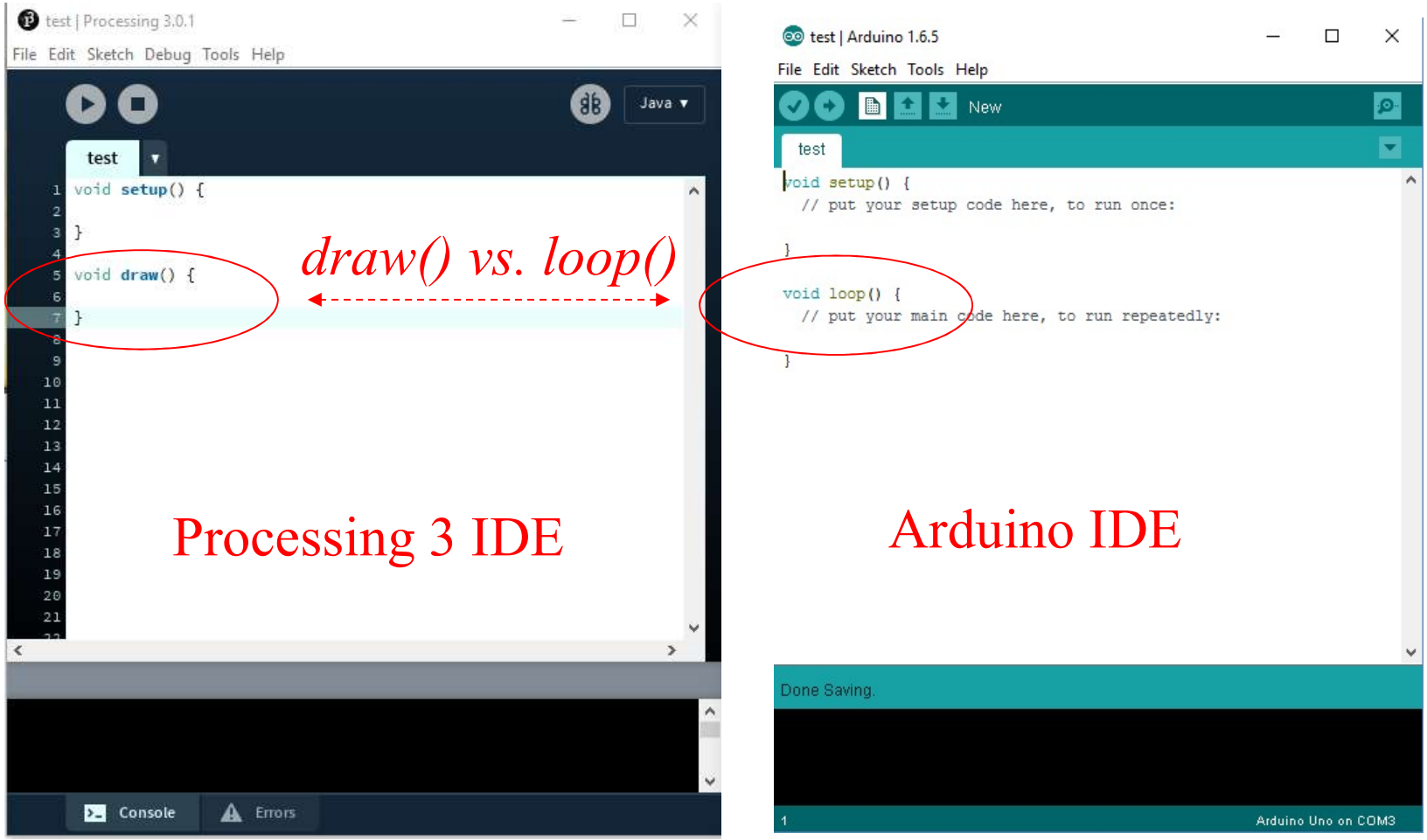
- "Processing is
  - an open source programming language and integrated development environment (IDE) [developed at the MIT Media Lab],
  - built for the electronic arts, new media art, and visual design communities
  - with the purpose of teaching the fundamentals of computer programming in a visual context,
  - and to serve as the foundation for electronic sketchbooks." [source: Wikipedia]



- The latest version is 3.2.3:

Processing (cont.)

- The Arduino IDE (in C++) is in fact a spawn out of the Processing project (in Java).



Processing (cont.)

- We will use Processing as a feature-rich user interface to graph the sensor data sent by Arduino on the serial port.
- Start with examining some of the examples. See https://processing.org/examples
- You will use Java (It has support for other languages too). See https://processing.org/tutorials/ to get familiar with coding in Processing.
- Its libraries extend its use beyond graphics and images into audio, video, and communication with other devices. (https://processing.org/reference/libraries/)

- For your own computers, you will need to download Processing from the provided link (free and available for many OSs).

MATLAB (Matrix Laboratory)

- MATLAB is a popular
  - high-level language and
  - interactive environment for numerical computing.
- It is a proprietary software, so you will need to use the lab computers to use MATLAB.

- Open-source alternatives you are allowed to use instead of MATLAB are Octave (https://www.gnu.org/software/octave/, installed on the lab computers) or SciLab (http://www.scilab.org/).
  - You may use SciLab or Octave if you decide, but be aware of syntax differences, limitation in source code availability, …
  - If you use Octave or SciLab (or other alternative software), do consult with the course instructor and you may demo and submit SciLab code for the MATLAB portion of the lab.

MATLAB

- Similarly, MATLAB is used to further process and plot the data. It reads the sensor data sent by Arduino on the serial port.
- Steps:
  1. Write the MATLAB code to read the sensor data values from the serial port.
  2. Use MATLAB to plot the data in real-time. One method is provided and explained in the next few slides (using StripChart.m) but you can use any alternative method.
- Create a MATLAB code file (.m file), to be submitted as a part of deliverables. Though for testing and initial implementation, you may simply type in your code using MATLAB's command line.
- The next slides provide more info.

MATLAB - Getting data

- Arduino constantly reads the sensor data and then provides it on the serial port. MATLAB can read that data from the serial port.

- One can use the *serial()* function in MATLAB to create a serial port object.
  - See: http://www.mathworks.com/help/matlab/ref/serial.html
- MATLAB treats the serial port object like a file, so one can use the usual functions (*fopen()*, *fscanf()*, *fclose(), …*) to access and read the data.
  - See:

  http://www.mathworks.com/help/matlab/ref/serial.fopen.html
  http://www.mathworks.com/help/matlab/ref/serial.fclose.html
  http://www.mathworks.com/help/matlab/ref/serial.fscanf.html

MATLAB - Getting data (cont.)

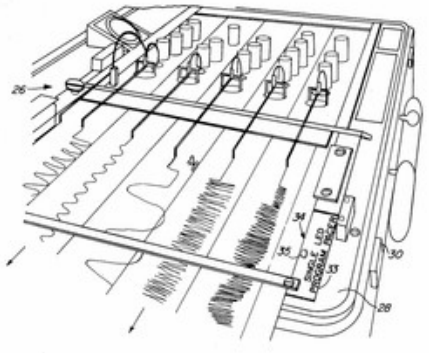
- The following is a sample code to receive a single value and to store it in the variable *result*.

```
s1 = serial('COM1', 'BaudRate', 9600, 'Parity', 'none', 'DataBits', 8, 'StopBits', 1, 'FlowControl', 'none');
fopen(s1);
val=fscanf(s1);
result = sscanf(val, '%f')
fclose(s1);
```

- You will need to change the code to constantly receive data and also later add the code to display in real-time. Note that
  - Adjust the serial port setting: e.g. change COM1 to whichever being used ...
  - You need repetition so you need to add a loop and include the code in red in it. This loop will eventually include both the code for reading the values and displacing it in real-time.
  - All code for initialization need to be called once at the top of your code. Those are the ones colour-coded green here. You need some simple modifications.
  - fclose needs to be called once at the end of your code.

MATLAB – Plotting

- We want to plot the data continuously in real-time.
  - Recall that MATLAB is constantly reading data from the serial port.
  - So, we need to display data in real-time, that is, while MATLAB is receiving this steam of data it will need to display it and constantly update the graph.
- You can use the code in the provided StripChart.m file (or StripChartXY.m).
  - Just for illustration and to give you an idea of what a strip chart is, consider the way a polygraph (lie detector) plots. It displays in real-time the data it receives, and moves the graph (by moving the paper in this case) to make room for the display of new data.



A chart recorder which is part of a polygraph

Image source: Wikipedia.org

Using StripChart to Plot in MATLAB

- Download the StripChart.m file. See the comments at the top of the StripChart.m file for the example; Simply use and modify the provided example:

```
%   Example:
    x = 1:1000;
    y = sin(2*pi*x/1000);
    hLine = plot(x,y);
    stripchart('Initialize',gca)
    for i=1:1000
    stripchart('Update',hLine,y(i))
    end
```

Initialization part of the stripchart code

Loop to display the stream of data

Replace y(i) with your own variable (for data you plot).

- You need it in the current or working directory.
  - The command pwd displays the current directory, e.g.:

    >> pwd
  - The command cd changes the current folder, e.g.:

    >> cd z:\my291\lab4

    See http://www.mathworks.com/help/techdoc/ref/cd.html

MATLAB – final remarks

- The final MATLAB code you write is quite simple and straightforward, given the provided examples.
  - You will need to modify and combine the two parts (as explained: getting data and displaying) to plot the data.
  - Allow the MATLAB code to complete (do not stop/break prematurely), otherwise fclose() will not be executed.
- Pay careful attention to the notes on the roll of each part of the code.
  - I have colour-coded them in the lecture notes to emphasize.
  - A colour-coding example: Green coloured code lines are mainly for setting up or initialization (of course, you do not need to colour code them in you report.)
  - Proper function use examples: *fopen*() or the *stripchart* initialization lines must be called only once at the beginning of the code. *fclose*() must be called only once at the end of your code.

Raspberry Pi 3

- Raspberry Pi is a series of credit-card single-board computers.
  - It has been around since 2012.
  - Different generations and models
    - Generations: 1, 2, 3, zero
    - Models: A, B, A+, B+
  - We will use the latest generation (released in Feb 2016).

https://en.wikipedia.org/wiki/Raspberry_Pi

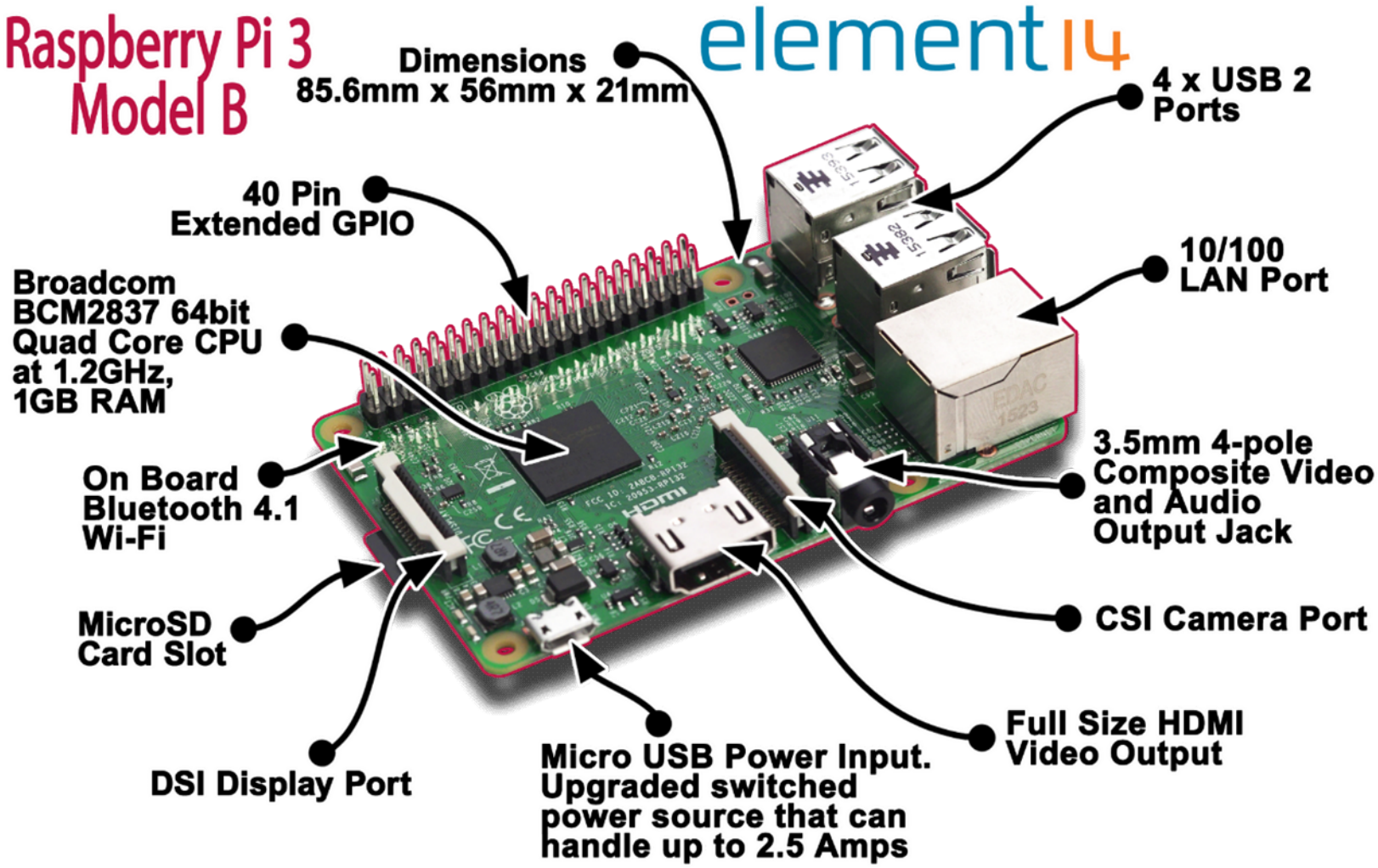
- It is commonly setup as an embedded Linux computing system.
  - Raspbian, Debian-based Linux optimized for Raspberry Pi, is officially provided by the Raspberry Pi foundation as the primary operating system.
    - We will use Raspbian as the primary OS.
  - Many other distributions and operating systems could be used on the Raspberry Pi.

Raspberry Pi 3 Specifications

- Like Pi 2, it has:
  - 1GB RAM
  - 4 USB ports
  - 40 GPIO pins
  - Full HDMI port
  - Ethernet port
  - Combined 3.5mm audio jack and composite video
  - Camera interface (CSI)
  - Display interface (DSI)
  - Micro SD card slot (now push-pull rather than push-push)
  - VideoCore IV 3D graphics core
- Compared to Pi 2, it has
  - A 1.2GHz 64-bit quad-core ARMv8 CPU
  - 802.11n Wireless LAN
  - Bluetooth 4.1

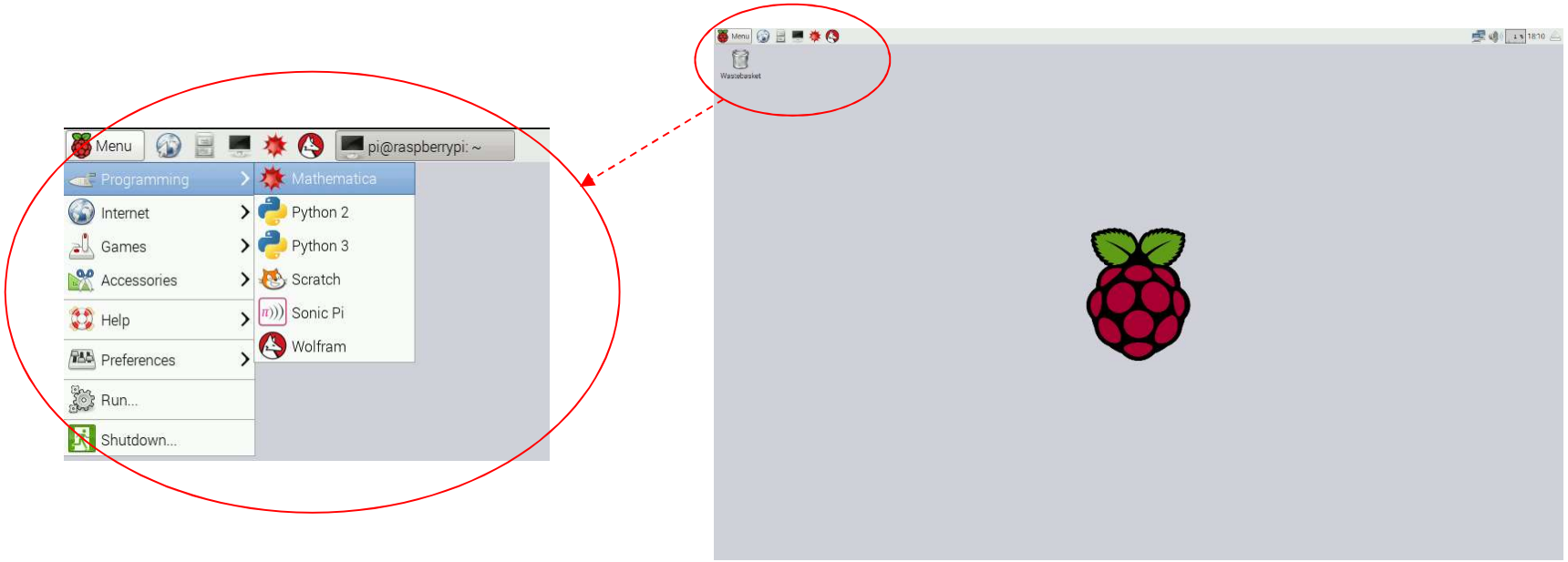
Raspberry Pi 3 – A Closer Look

- The figure shows different parts of the Pi:



Raspbian

- Our Raspberry Pi 3 comes with a 16GB microSD pre-loaded with Raspbian (via NOOB, https://www.raspbian.org/).
- The image shows the basic GUI. The look, of course, could be different depended of the distribution.

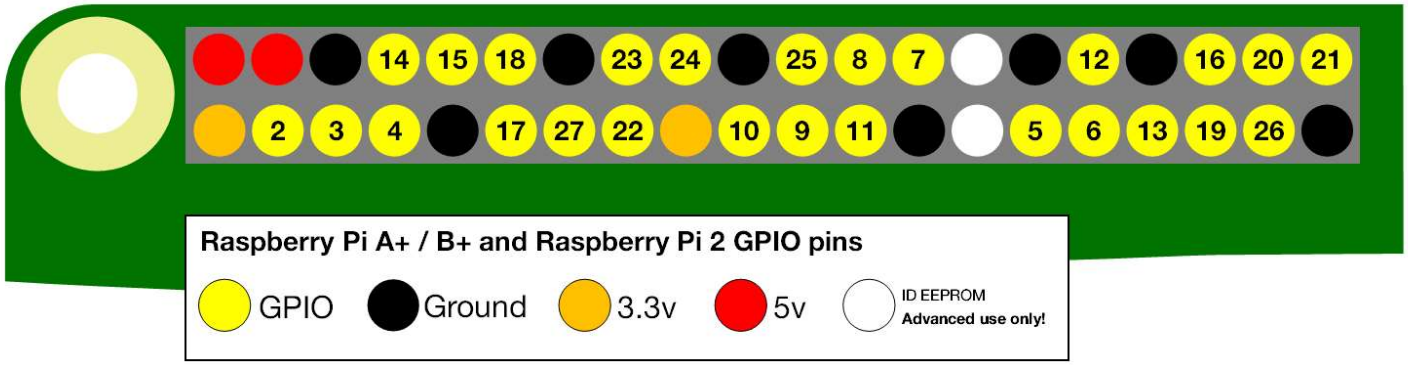


- A part of the Raspbian distribution, many applications come pre-installed, and one can easily add more.

Raspberry Pi 3 GPIO

- The provided Raspberry Pi 3 has a 40-pin connecter (Note that some earlier PI models are 26-pin).
  - GPIO: general purpose input/output

- The figure identifies the pins:

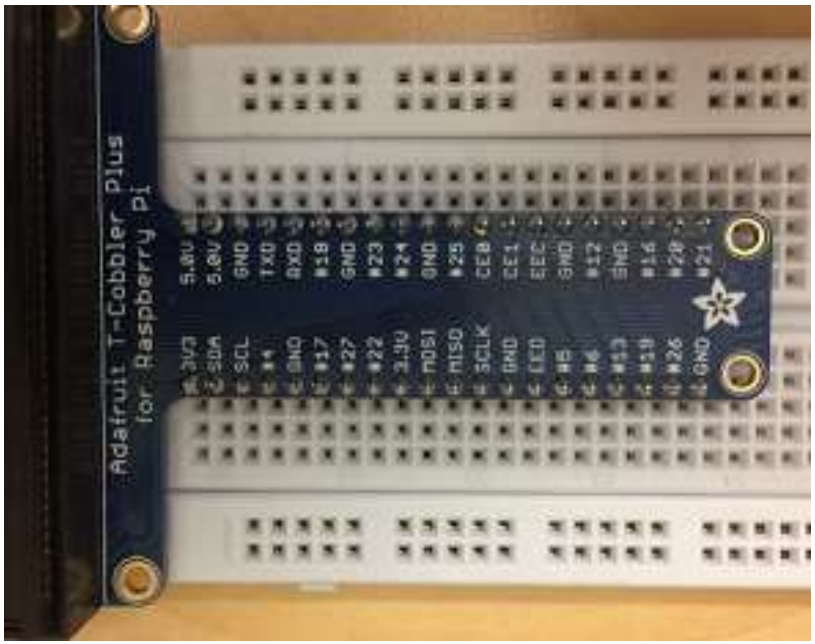


Source: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/

- Whenever you are using the T-Cobbler (next slide), you can conveniently use the labels on it to identify the pins instead.

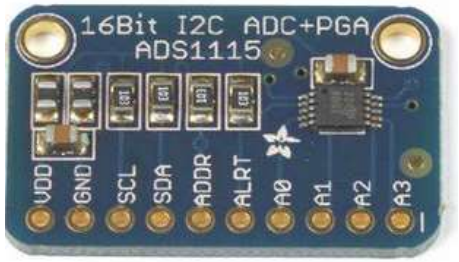
T-Cobbler and Cable

- A rather fancy T-shape cobbler (and cable) is included in your kit to make the Pi's GPIO easily accessible on the breadboard.
- The T-shape makes it a lot easier to read the labels, but it is not compact. The figures show one way of conveniently placing it at the edge of the breadboard.



Raspberry Pi 3 - Analog

- Note that unlike Arduino, there is no analog input pin readily available on the Pi.
- One will need to use an analog to digital converter (ADC) chip or breakout for that purpose, examples:



- You will not use analog input in lab 4 or lab 5 using the Pi, so there is no concern here.

- See here for more info:
  - https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/overview

Python 3

- We will use Python 3 to program on the Pi.
- Python 3 and Idle (an IDE that comes with Python) already come pre-installed on Raspbian.
- You do not need to fully learn Python now, just enough to do the exercises in Lab 4 and Lab 5.
  - Python is rather different from C/Java, but it would be quite straightforward to follow the provided tutorial.
- See https://www.raspberrypi.org/learning/physical-computing-guide/test-led-python/
- For example:
  - Use *import* to import the needed modules (for example, time (for delay) and GPIO (for Pi's GPIO)).
  - Use the library functions (this time in python and for the PI) to set up the pins you are using, and to read or write from them.
  - …

Python 2 vs. Python 3

- Be aware of the differences between Python 2 and Python 3 (https://docs.python.org/3.0/whatsnew/3.0.html).
- Python 3 is not backward compatible.

- For example, the print statement is used differently:

```
print "LED off"          # Python 2, print statement

Vs.

print( “LED off”)        # Python 3, as a function
```

Some Notes

- Be aware of what GPIO module your are using.
  - Also for RPI.GPIO module, one may use either of the BOARD or BCM modes.
    - Note that, for example, GPIO #25 is not pin #25.
  - ***BCM mode***: based on GPIO numbering scheme
    - GPIO.setmode(GPIO.BCM)
  - ***BOARD mode***: based on pin numbering scheme
    - GPIO.setmode(GPIO.BOARD)

- You should get comfortable with using the Linux terminal (command line). That may make you life easier for some tasks like installing new applications, update/upgrade Linux, …

References

- Arduino Reference:
  http://arduino.cc/en/Reference/HomePage

- Processing:
  Download website: https://processing.org/download/
  Tutorial: https://processing.org/tutorials/
  Reference: https://processing.org/reference/

- MATLAB tutorial:
  http://www.mathworks.com/academia/student_center/tutorials/launchpad.html

- Raspberry Pi: https://www.raspberrypi.org/

- Graph: http://arduino.cc/en/Tutorial/Graph

- See the datasheets posted on Connect