Learning Lab:

Raspberry Pi STEM Kit

By Lt Col Jonathan Lartigue

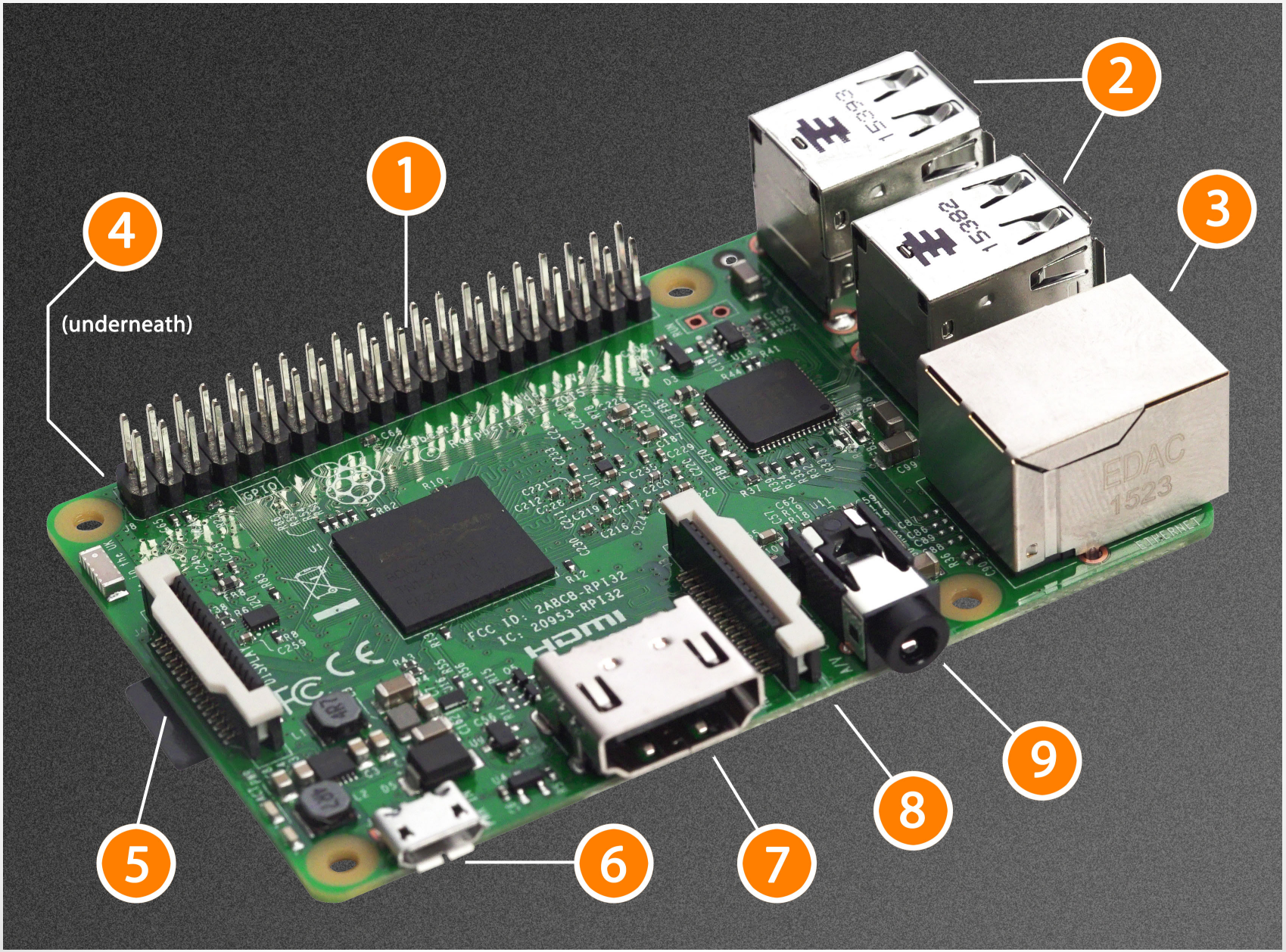
2017 National Conference

San Antonio, Texas • 2 September 2015

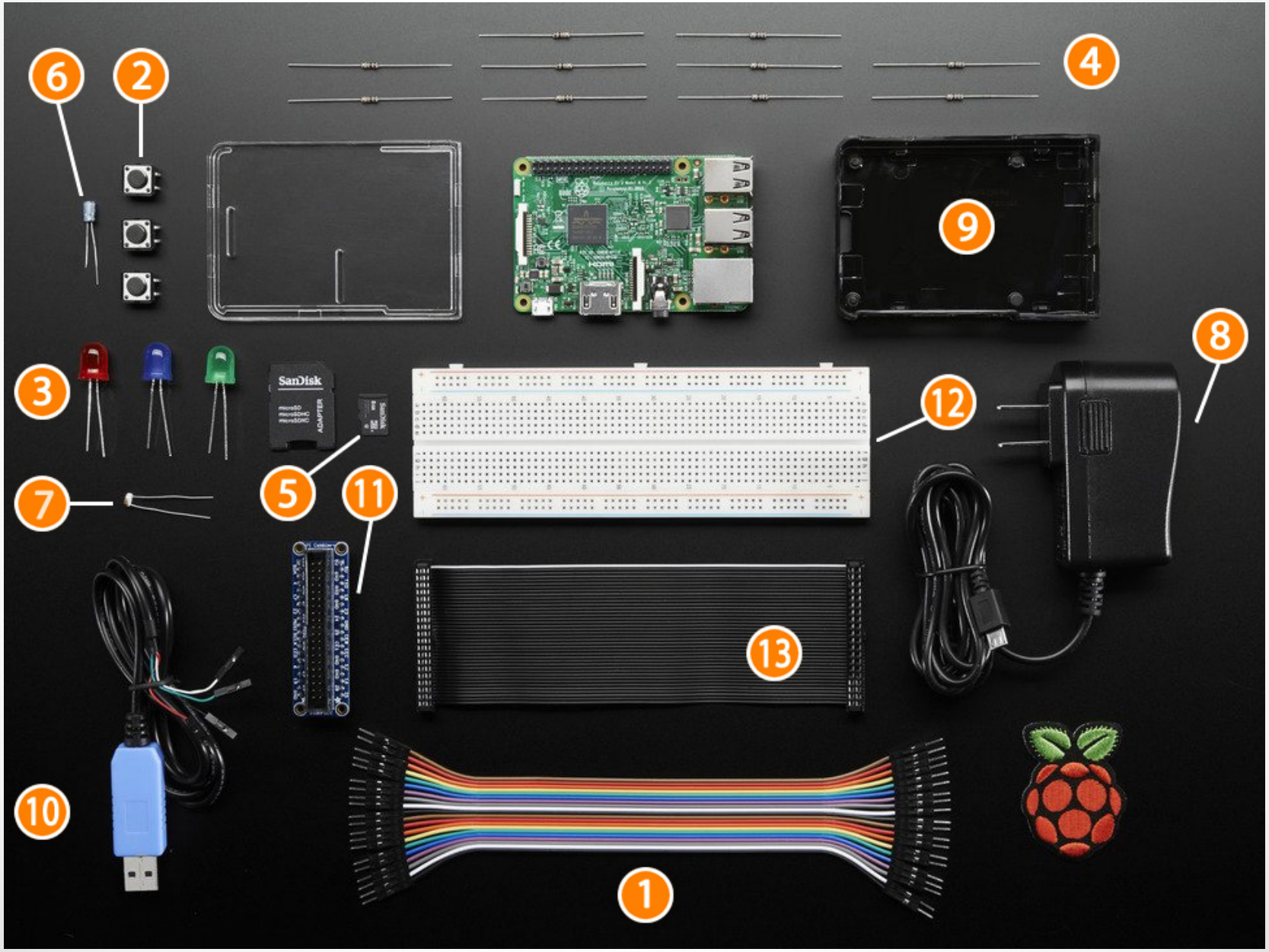
Topics Covered

- Introduction
- Kit Components
- Workbook Modules
- Learning Lab Exercise

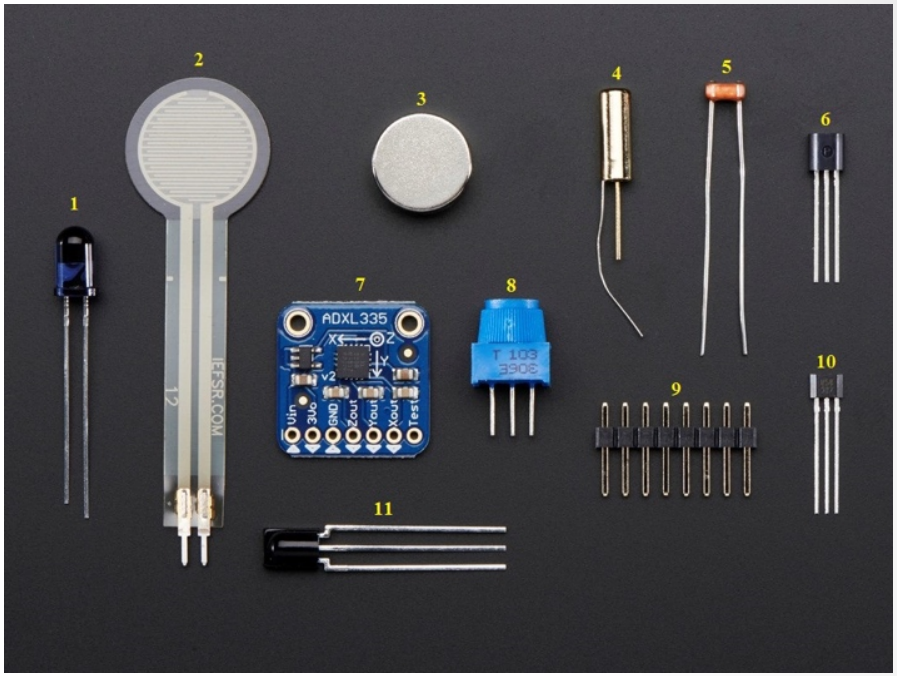
The Raspberry Pi Device



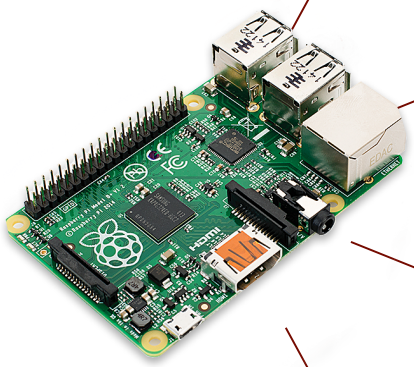
Core RaspPi Starter Kit



Add-On Sensor Packs



Proposed STEM Kit Contents



Add-on Kit 1

- Anemometer
- Temperature, Humidity, & Pressure Sensors

Add-on Kit 2

- Real-time clock
- LCD and Keypad
- GPS Chip

Add-on Kit 3

- Color Touchscreen
- Pulse Sensor

Add-on Kit 4

- Sensor Pack 900
- Camera Board

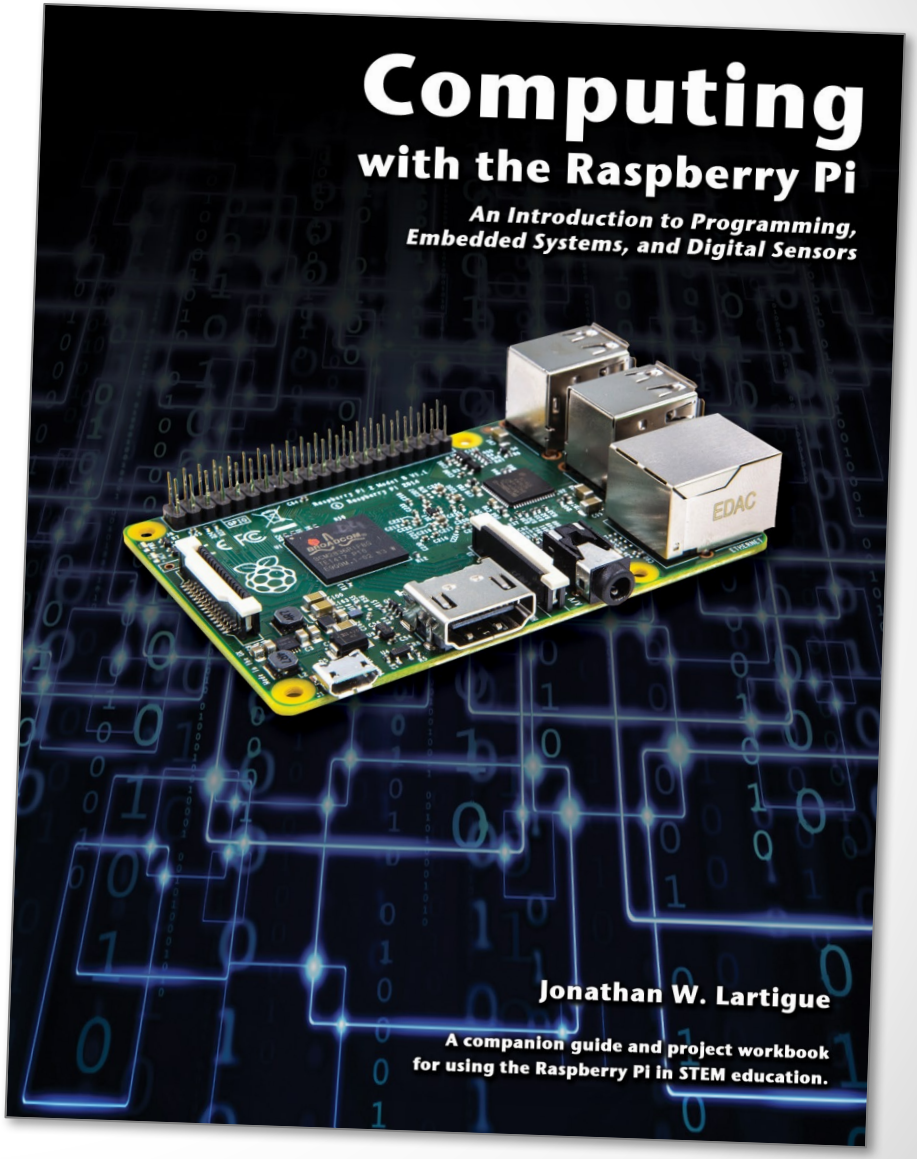
Proposed STEM Kit Costs

Item	Part # [1]	Estimated Cost [1]
Core Kit		**$90 *(entire kit)***
Raspberry Pi 1 Model B+ Starter Pack, incl. Raspberry Pi	#2125	$89.95
Add-on Kit #1		**$74 *(entire kit)***
Anemometer Wind Speed Sensor w/Analog Voltage Output	#1733	$44.95
AM2302 Temperature-Humidity Sensor	#393	$15.00
BMP180 Barometric Pressure/Temperature/Altitude Sensor	#1603	$9.95
MCP3008 8-Channel 10-Bit ADC With SPI Interface	#856	$3.75
Add-on Kit #1		**$68 *(entire kit)***
DS1307 Real Time Clock Breakout Board Kit	#264	$9.00
Adafruit Blue & White 16x2 LCD and Keypad Kit for Raspberry Pi	#1115	$19.95
Adafruit Ultimate GPS Breakout, Version 3	#746	$39.95
Add-on Kit #3		**$63 *(entire kit)***
PiTFT Assembled 320x240 2.8" TFT+Touchscreen for Raspberry Pi	#1601	$34.95
Pulse Sensor Amped	#1093	$25.00
MCP3008 8-Channel 10-Bit ADC With SPI Interface	#856	$3.75
Add-on Kit #4		**$69 *(entire kit)***
Sensor Pack 900	#176	$35.00
Raspberry Pi Camera Board	#1367	$29.95
MCP3008 8-Channel 10-Bit ADC With SPI Interface	#856	$3.75

[1] *When supplied from Adafruit.com, as of June 2015.*

Workbook Exercises

- The Core Kit and four Add-On kits will support 15 pre-defined projects.
  - Each activity module comes with complete instructions, source code, and wiring diagrams.
- Modules will be distributed as a PDF copy of an unpublished manuscript.
  - Permission from the author for use within Civil Air Patrol will be provided.



Included Lessons

Skill Level
BEGINNER

Exercises at this skill level are within the capabilities of any beginning programmer. Little or no prior experience with programming or the Raspberry Pi is necessary to complete these exercises.

Skill Level
NOVICE

Novice exercises presume some previous exposure to programming and/or the Raspberry Pi. Satisfactory completion of the Getting Started guide and Beginner level exercises should adequately prepare students to exercises at the Novice level.

Skill Level
INTERMEDIATE

Intermediate level exercises require a more thorough foundation in programming and working with the Raspberry Pi. Exercises at this level are more time consuming and may require greater programming skills, more complex interaction with digital circuits or sensors, and the ability to utilize third party tools or programming interfaces.

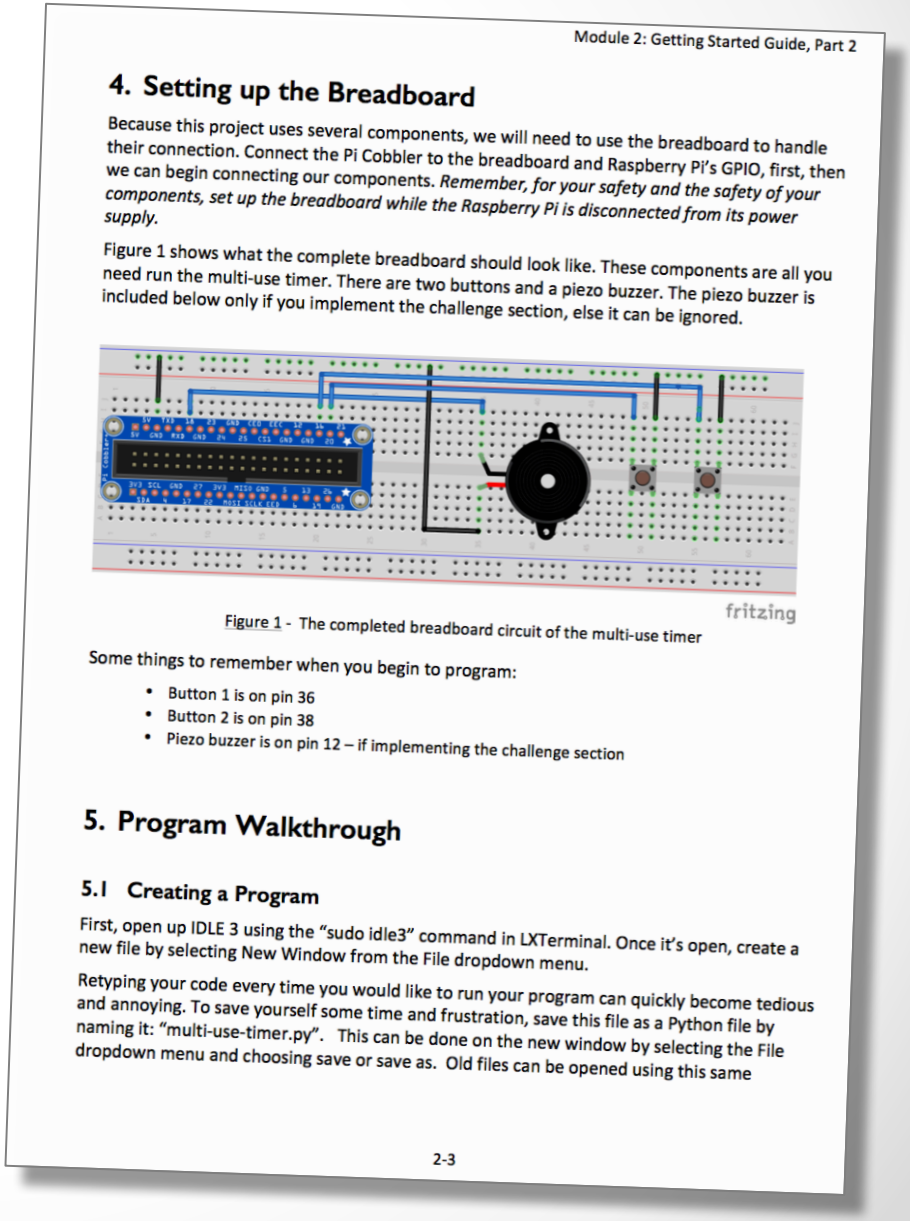
Skill Level
ADVANCED

Advanced exercises will require a significant time investment, advanced programming skills, and the ability to work with multiple, complex digital circuits and sensors. Some assembly and/or soldering of digital components may be required.

Included Lessons

Supported Projects

- Three Getting Started Guides
  - Getting Started Guide, Part 1
    - Introduction to Raspberry Pi, kit contents, and digital circuits
  - Getting Started Guide, Part 2
    - Working with the breadboard and electronic components.
    - Introduction to programming with Python
  - Getting Started Guide, Part 3
    - Advanced Topics



Module 2: Getting Started Guide, Part 2

4. Setting up the Breadboard

Because this project uses several components, we will need to use the breadboard to handle their connection. Connect the Pi Cobbler to the breadboard and Raspberry Pi's GPIO, first, then we can begin connecting our components. *Remember, for your safety and the safety of your components, set up the breadboard while the Raspberry Pi is disconnected from its power supply.*

Figure 1 shows what the complete breadboard should look like. These components are all you need run the multi-use timer. There are two buttons and a piezo buzzer. The piezo buzzer is included below only if you implement the challenge section, else it can be ignored.

fritzing

Figure 1 - The completed breadboard circuit of the multi-use timer

Some things to remember when you begin to program:

- Button 1 is on pin 36
- Button 2 is on pin 38
- Piezo buzzer is on pin 12 – if implementing the challenge section

5. Program Walkthrough

5.1 Creating a Program

First, open up IDLE 3 using the "sudo idle3" command in LXTerminal. Once it's open, create a new file by selecting New Window from the File dropdown menu.

Retyping your code every time you would like to run your program can quickly become tedious and annoying. To save yourself some time and frustration, save this file as a Python file by naming it: "multi-use-timer.py". This can be done on the new window by selecting the File dropdown menu and choosing save or save as. Old files can be opened using this same

2-3

Included Projects

- Three Beginner Exercises
  - Keypad Security Lock
  - Reaction-Time Simulator Game
  - Heart Rate Monitor

Skill Level
BEGINNER

- Three Novice Exercises
  - Morse Code Generator
  - 'Simon Says' Game
  - Sensor Pack Mini-Labs (four mini exercises)

Skill Level
NOVICE

Included Projects

- Four Intermediate Exercises
  - World Clock
  - Tic-Tac-Toe Game (with touchscreen)
  - Turbulence and Earthquake Monitor
  - Home Security Monitor

Skill Level
INTERMEDIATE

- Two Advanced Exercises
  - Build-Your-Own Airport Weather Station
  - Build-Your-Own GPS System

Skill Level
ADVANCED

Learning Lab Exercise

- Hello, World!

Learning Lab Exercise

```
# Hello, RasPi!
# file: hello.py

print(“Hello, 2017 CAP
National Conference
Attendees!”)
```

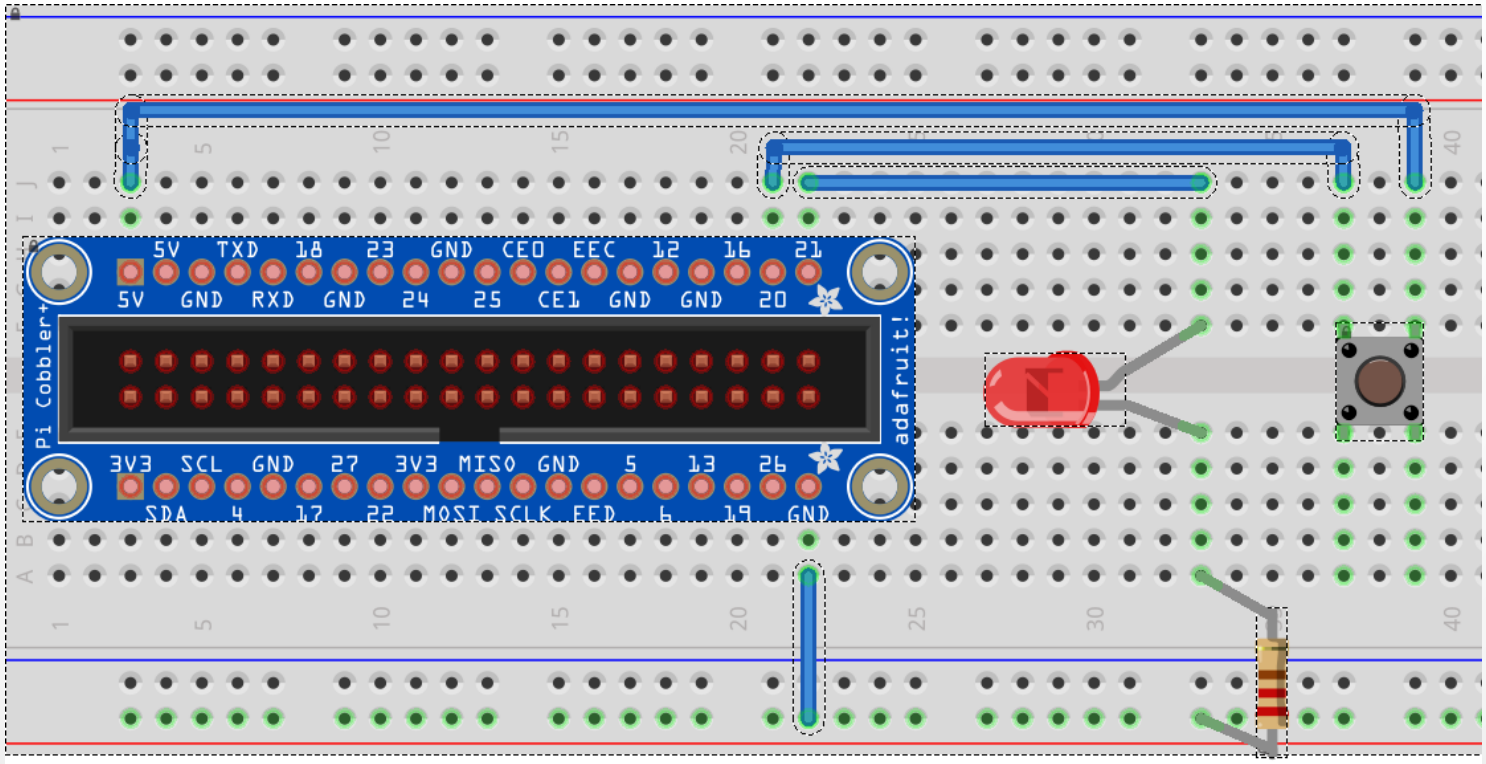
```
// Hello, RasPi
// file: hello.c

#include <stdio.h>


int main(void)
{
  printf("Hello, 2017
CAP National Conference
Attendees!\n");
  return 0;
}
```

Blinky: Wire the Breadboard

- You'll need:
  - RasPi Cobbler Adapter
  - A Light Emitting Diode (LED)
  - One Resistor
  - One Push Button



Blinky: Write the Code (v1.0)

```
# RasPi Blinky Exercise, Version 1
# file: blinky1.py

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(40,GPIO.OUT)

for i in range(50):
   GPIO.output(40,True)
   time.sleep(1)
   GPIO.output(40,False)
   time.sleep(1)

GPIO.cleanup()
```

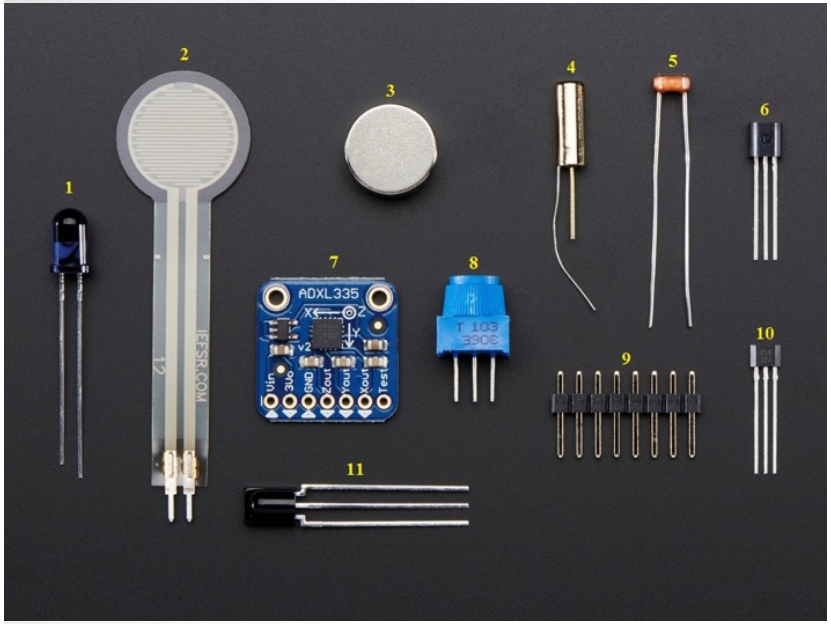
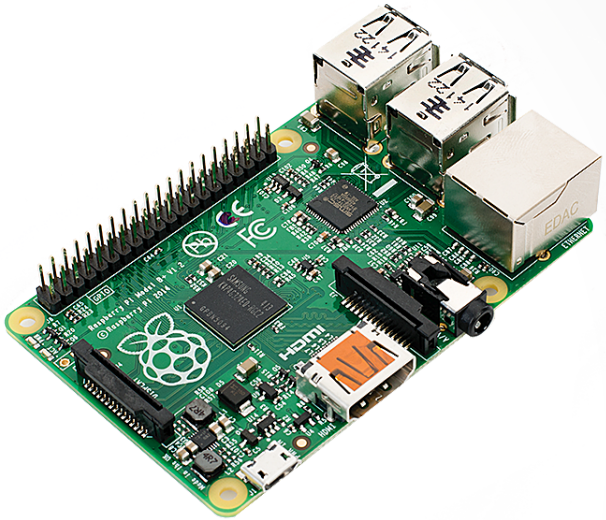
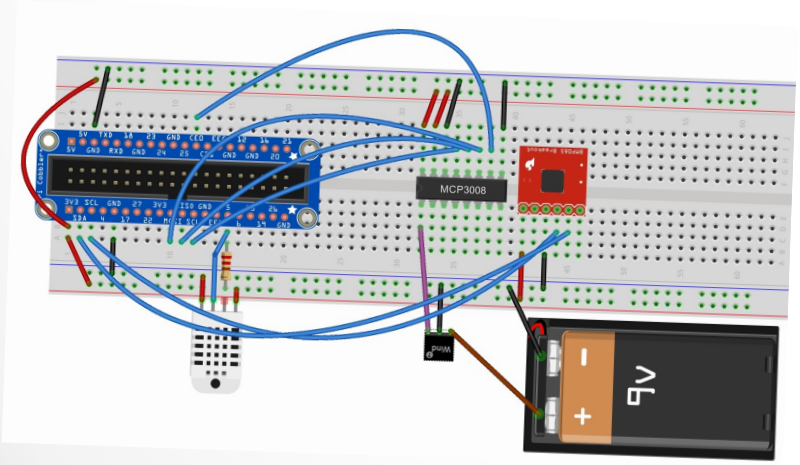
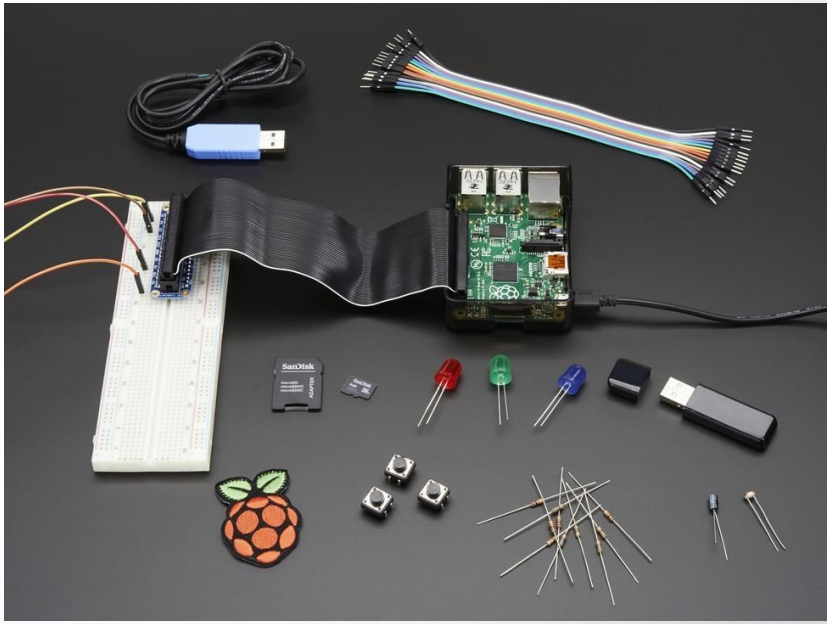
Blinky: Write the Code (v2.0)

```
# RasPi Blinky Exercise, Version 2
# file: blinky2.py

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(40,GPIO.OUT)
GPIO.setup(38,GPIO.IN, pull_up_down =
GPIO.PUD_DOWN)

# This is an endless loop, not very elegant
while True:
if (GPIO.input(38) == 1):
      GPIO.output(40,True)
   else:
      GPIO.output(40,False)

GPIO.cleanup()
```

Questions?