Imperial College London

Tutorial 6

The Challenges

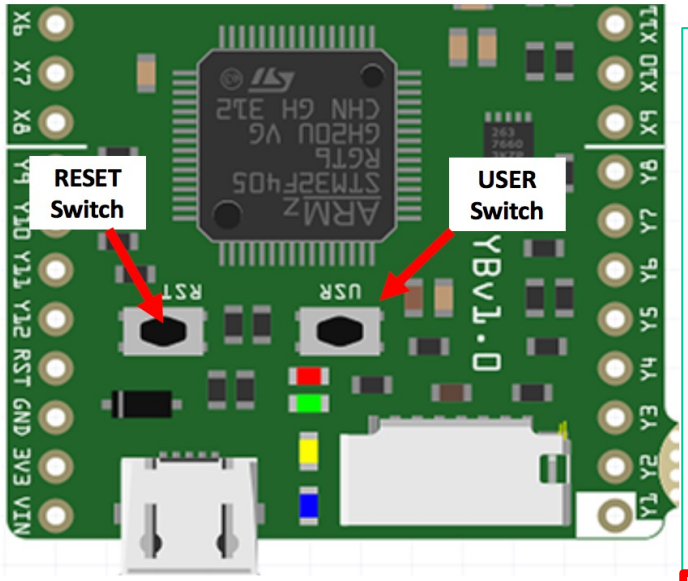
Prof Peter YK Cheung

Dyson School of Design Engineering

URL: www.ee.ic.ac.uk/pcheung/teaching/DE2_EE/
E-mail: p.cheung@imperial.ac.uk

Tips 1: From Idling to Running

- An effective way to run the Segway:

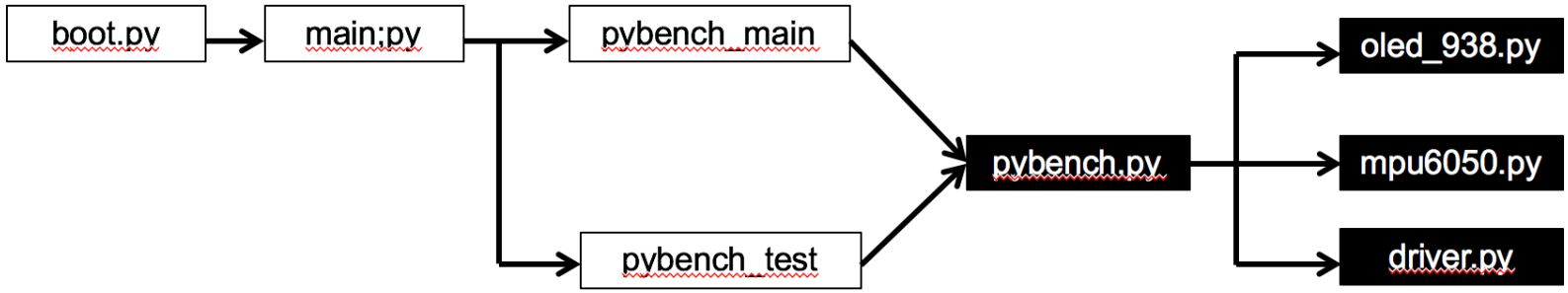



```
# Use OLED to say what Segway is doing
oled = OLED_938(pinout={'sda': 'Y10', 'scl': 'Y9', 'res': 'Y8'},
        height=64, external_vcc=False, i2c_devid=61)
oled.poweron()
oled.init_display()
oled.draw_text(0,0, 'Group xx')
oled.draw_text(0,10, 'Milestone 1: Driving')
oled.draw_text(0,20, 'Press USR button')
oled.display()

print('Performing Milestone 1')
print('Waiting for button press')
trigger = pyb.Switch()  # create trigger switch object
while not trigger():    # wait for trigger pressed
    time.sleep(0.001)
while trigger(): pass   # wait for release
print('Button pressed - Running')

# ....   rest of program
```

Tips 2: Previous main.py

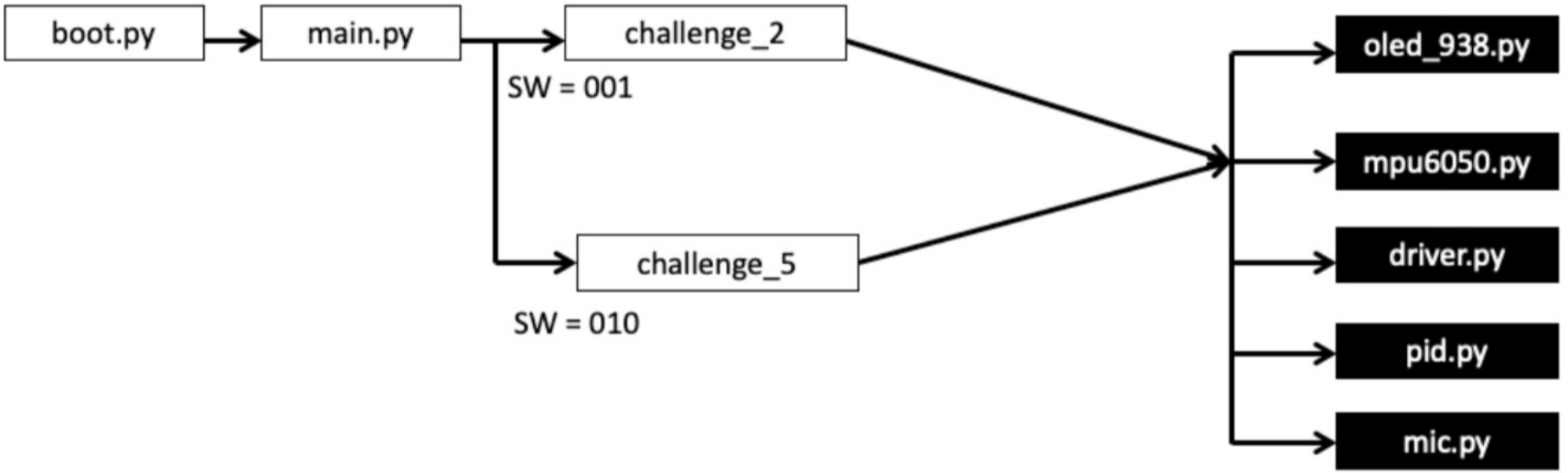



```
import pyb
from pyb import Pin, LED

#  Configure X2:4, X7 as setting input pin -
s0=Pin('Y3',pyb.Pin.IN,pyb.Pin.PULL_UP)
s1=Pin('X6',pyb.Pin.IN,pyb.Pin.PULL_UP)
r_LED = LED(1)
g_LED = LED(2)
y_LED = LED(3)
b_LED = LED(4)
'''
Define various test functions
'''
def read_sw():
    value = 3 - (s0.value() + 2*s1.value())
    if (not s0.value()):
        y_LED.on()
    if (not s1.value()):
        g_LED.on()
    return value
```

```
if read_sw() == 0:
    print('Running PyBench')
    execfile('pybench_main.py')
elif read_sw() == 1:
    print('Running PyBench Selftest')
    execfile('pybench_test.py')
elif read_sw() == 2:
    print('Running PyBench Selftest')
    execfile('pybench_test.py')
elif read_sw() == 3:
    print('Running User Python Program')
    execfile('user.py')
```

Tips 2: New main.py



Tips 3: Pseudo-code for Challenge 4

```
#  Import relevant packages that you use ...
import pyb
from pyb import Pin, Timer, ADC, DAC, LED
import micropython   # Needed for interrupt
micropython.alloc_emergency_exception_buf(100)
.........

#  Initialise various peripherals e.g. OLED, IMU etc

#  Initialise different constants, variables, arrays etc

#  Read the dancing steps from file into array

#  Wait for USR switch pressed

tic = pyb.millis()          # mark time now in msec

try:              # Try to handle exception
    while True: # always have infinite loop
        if buffer is full, detect beat
        if beat detected, do next dance move

finally:          # always executed if exception
    motor.A_stop()
    motor.B_stop()
```

Tips 4: PITCH ANGLE ESTIMATION

```
# Pitch angle calculation using complementary filter
def pitch_estimate(pitch, dt, alpha):
    theta = imu.pitch()
    pitch_dot = imu.get_gy()
    pitch = alpha*(pitch + pitch_dot*dt) + (1-alpha)*theta
    return (pitch, pitch_dot)
```

```
tic1 = pyb.micros()

while True:      # infinite loop
    dt = pyb.micros() - tic1
```

Tips 5: PID CONTROLLER

$$w(t) = K_p e(t) + K_d \dot{e}(t) + K_i \int e(\tau) d\tau$$

Input to function: pitch angle, (rate of change of pitch (pitch_dot), target (or set-point), cumulative pitch error (integral term).

Output of the function: PWM drive value limited to ±100.

```
#  Calculate the pitch error pError

#  Computer the output W as sum of P, I and D terms

#  Accumulate the pError term for integration

#  Limit the output W to ±100

#  Return the W drive value
```

Tips 6: TUNING THE PID CONTROLLER

```
trigger = pyb.Switch()
scale = 2.0
while not trigger(): # wait to tune Kp
    time.sleep(0.001)
    K_p = pot.read() * scale / 4095 # use pot to set Kp
    oled.draw_text(0, 30, 'Kp = {:5.2f}'.format(K_p)) # display live value on oled
    oled.display()
while trigger(): pass # wait for button release
while not trigger(): # wait to tune Ki
    time.sleep(0.001)
    K_i = pot.read() * scale / 4095 # use pot to set Ki
    oled.draw_text(0, 40, 'Ki = {:5.2f}'.format(K_i)) # display live value on oled
    oled.display()
while trigger(): pass # wait for button release
while not trigger(): # wait to tune Kd
    time.sleep(0.001)
    K_d = pot.read() * scale / 4095 # use pot to set Kd
    oled.draw_text(0, 50, 'Kd = {:5.2f}'.format(K_d)) # display live value on oled
    oled.display()
while trigger(): pass # wait for button release

print('Button pressed. Running script.')
oled.draw_text(0, 20, 'Button pressed. Running.')
oled.display()

... the rest of the program ....
```

Tips 7: PSEUDO-CODE FOR SELF-BALANCING, DANCING SEGWAY

```
#  Import relevant packages that you use ...
import pyb
from pyb import Pin, Timer, ADC, DAC, LED
import micropython   # Needed for interrupt
micropython.alloc_emergency_exception_buf(100)
........

#  Initialise various peripherals e.g. OLED, IMU etc

#  Initialise different constants, variables,

#  Read the dancing steps from file into array

#  Use Potentiometer and USR switch to tune Kp

tic2 = pyb.millis()         # mark time now in
```

```
try:            # Try to handle exception
    tic1 = pyb.micros()
    while True:     # infinite loop
        dt = pyb.micros() - tic1
        if (dt > 5000):     # loop period is 5 msec or 200Hz
            estimate pitch angle and pitch_dot
            update tic1
            do PID control
            use returned value to move motor
        if microphone buffer is full
            test if beat has occurred
            if yes, update the target of pitch angle
            update tic2

finally:        # always executed if exception
    motor.A_stop()
    motor.B_stop()
```