**from machine import Pin, PWM, ADC, Timer**

**import time**

**# Set up pins**

**led\_pin = Pin(2, Pin.OUT)**

**pwm\_pin = PWM(Pin(5))**

**square\_wave\_pin1 = Pin(16, Pin.IN)**

**square\_wave\_pin2 = Pin(17, Pin.IN)**

**analog\_pin = ADC(Pin(34))**

**analog\_pin.atten(ADC.ATTN\_11DB) # Set attenuation to 11 dB for 0-3.3V range**

**# Task 1: Output digital signal**

**def task1(timer):**

**pwm\_pin.duty(512)**

**time.sleep\_us(200)**

**pwm\_pin.duty(0)**

**time.sleep\_us(50)**

**pwm\_pin.duty(512)**

**time.sleep\_us(200)**

**pwm\_pin.duty(0)**

**time.sleep\_us(50)**

**pwm\_pin.duty(0)**

**time.sleep\_us(50)**

**pwm\_pin.duty(512)**

**time.sleep\_us(200)**

**pwm\_pin.duty(512)**

**time.sleep\_us(30)**

**pwm\_pin.duty(0)**

**time.sleep\_us(50)**

**pwm\_pin.duty(512)**

**# Task 2: Measure frequency of square wave signal**

**def task2(timer):**

**# Measure frequency**

**start\_time = time.ticks\_us()**

**while square\_wave\_pin1.value() == 0:**

**pass**

**while square\_wave\_pin1.value() == 1:**

**pass**

**period = time.ticks\_diff(time.ticks\_us(), start\_time)**

**frequency = 1000000 / period**

**# Send frequency to task 5**

**task5\_values[0] = int(frequency)**

**# Task 3: Measure frequency of another square wave signal**

**def task3(timer):**

**# Measure frequency**

**start\_time = time.ticks\_us()**

**while square\_wave\_pin2.value() == 0:**

**pass**

**while square\_wave\_pin2.value() == 1:**

**pass**

**period = time.ticks\_diff(time.ticks\_us(), start\_time)**

**frequency = 1000000 / period**

**# Send frequency to task 5**

**task5\_values[1] = int(frequency)**

**# Task 4: Read and filter analog input**

**def task4(timer):**

**# Read analog input**

**reading = analog\_pin.read()**

**# Filter reading**

**filtered\_values.append(reading)**

**if len(filtered\_values) > 4:**

**filtered\_values.pop(0)**

**average\_analog\_in = sum(filtered\_values) / len(filtered\_values)**

**# Check for error and set LED accordingly**

**if average\_analog\_in > 1650: # half of maximum range is 1650**

**led\_pin.on()**

**else:**

**led\_pin.off()**

**# Task 5: Log information to serial port**

**def task5(timer):**

**# Send data to serial port**

**frequency1 = task5\_values[0]**

**frequency2 = task5\_values[1]**

**print("%d,%d" % (frequency1, frequency2))**

**import machine**

**import time**

**# Initialize pins for digital output and error LED**

**digital\_out = machine.Pin(5, machine.Pin.OUT)**

**error\_led = machine.Pin(19, machine.Pin.OUT)**

**# Initialize analog input pin**

**analog\_in = machine.ADC(0)**

**# Initialize variables for frequency measurement**

**task2\_timer = machine.Timer(0)**

**task2\_count = 0**

**task2\_frequency = 0**

**task3\_timer = machine.Timer(1)**

**task3\_count = 0**

**task3\_frequency = 0**

**# Define function for Task 1**

**def task1():**

**digital\_out.on()**

**time.sleep\_us(200)**

**digital\_out.off()**

**time.sleep\_us(50)**

**digital\_out.on()**

**time.sleep\_us(200)**

**digital\_out.off()**

**time.sleep\_us(50)**

**digital\_out.off()**

**time.sleep\_us(50)**

**digital\_out.on()**

**time.sleep\_us(200)**

**digital\_out.on()**

**time.sleep\_us(30)**

**digital\_out.off()**

**time.sleep\_us(50)**

**digital\_out.on()**

**# Define function for Task 2 (frequency measurement)**

**def task2(timer):**

**global task2\_count, task2\_frequency**

**task2\_frequency = task2\_count / 20 # calculate frequency in Hz**

**task2\_count = 0 # reset counter**

**print("Task 2 frequency: ", task2\_frequency)**

**# Define function for Task 3 (frequency measurement)**

**def task3(timer):**

**global task3\_count, task3\_frequency**

**task3\_frequency = task3\_count / 8 # calculate frequency in Hz**

**task3\_count = 0 # reset counter**

**print("Task 3 frequency: ", task3\_frequency)**

**# Define function for Task 4 (analog input reading and filtering)**

**def task4():**

**global error\_led**

**readings = [0, 0, 0, 0]**

**for i in range(4):**

**readings[i] = analog\_in.read() # read analog input**

**average\_analog\_in = sum(readings) / 4 # calculate average**

**if average\_analog\_in > 1650: # check for error condition**

**error\_led.on()**

**else:**

**error\_led.off()**

**# Define function for Task 5 (logging to serial port)**

**def task5():**

**print("%d,%d" % (task2\_frequency, task3\_frequency))**

**# Initialize cyclic executive with tasks and timings**

**tasks = [task1, (task2\_timer.init(period=20, mode=machine.Timer.PERIODIC, callback=task2),),**

**(task3\_timer.init(period=8, mode=machine.Timer.PERIODIC, callback=task3),),**

**task4, (machine.Timer(-1).init(period=100, mode=machine.Timer.PERIODIC, callback=task5),)]**

**timings = [4, 20, 8, 20, 100]**

**# Start cyclic executive**

**last\_time = time.ticks\_ms()**

**task\_index = 0**

**while True:**

**current\_time = time.ticks\_ms()**

**elapsed\_time = time.ticks\_diff(current\_time, last\_time)**

**if elapsed\_time >= timings[task\_index]:**

**task = tasks[task\_index]**

**if isinstance(task, tuple):**

**task[0]()**

**else:**

**task()**

**last\_time = current\_time**

**task\_index = (task\_index + 1) % len(tasks)**

**import machine**

**import time**

**# Task 1 - Output a digital signal**

**def task1():**

**digital\_out\_pin = machine.Pin(12, machine.Pin.OUT)**

**digital\_out\_pin.value(1)**

**time.sleep\_us(200)**

**digital\_out\_pin.value(0)**

**time.sleep\_us(50)**

**digital\_out\_pin.value(1)**

**time.sleep\_us(200)**

**digital\_out\_pin.value(0)**

**time.sleep\_us(50)**

**digital\_out\_pin.value(0)**

**time.sleep\_us(50)**

**digital\_out\_pin.value(1)**

**time.sleep\_us(200)**

**digital\_out\_pin.value(1)**

**time.sleep\_us(30)**

**digital\_out\_pin.value(0)**

**time.sleep\_us(50)**

**digital\_out\_pin.value(1)**

**# Task 2 - Measure the frequency of a square wave signal**

**def task2():**

**frequency\_measurement\_pin = machine.Pin(13, machine.Pin.IN)**

**pulse\_count = 0**

**start\_time = time.ticks\_ms()**

**while time.ticks\_diff(time.ticks\_ms(), start\_time) < 20:**

**pulse\_count += frequency\_measurement\_pin.value()**

**frequency = pulse\_count \* 50 # assuming 50% duty cycle**

**print("Frequency 1: %d" % frequency)**

**return frequency**

**# Task 3 - Measure the frequency of a second square wave signal**

**def task3():**

**frequency\_measurement\_pin = machine.Pin(14, machine.Pin.IN)**

**pulse\_count = 0**

**start\_time = time.ticks\_ms()**

**while time.ticks\_diff(time.ticks\_ms(), start\_time) < 8:**

**pulse\_count += frequency\_measurement\_pin.value()**

**frequency = pulse\_count \* 125 # assuming 50% duty cycle**

**print("Frequency 2: %d" % frequency)**

**return frequency**

**# Task 4 - Read one analogue input, and compute a filtered analogue value**

**def task4():**

**analog\_in\_pin = machine.ADC(4)**

**readings = [0, 0, 0, 0]**

**sum\_of\_readings = 0**

**for i in range(4):**

**readings[i] = analog\_in\_pin.read()**

**sum\_of\_readings += readings[i]**

**average\_analog\_in = sum\_of\_readings / 4**

**if average\_analog\_in > 1650: # half of 3.3V maximum range**

**error\_led\_pin = machine.Pin(15, machine.Pin.OUT)**

**error\_led\_pin.value(1)**

**else:**

**error\_led\_pin = machine.Pin(15, machine.Pin.OUT)**

**error\_led\_pin.value(0)**

**return average\_analog\_in**

**# Task 5 - Log frequency measurements to serial port**

**def task5():**

**frequency1 = task2()**

**frequency2 = task3()**

**print("%d,%d" % (frequency1, frequency2))**

**# Initialize cyclic executive**

**tasks = [(task1, 4), (task2, 20), (task3, 8), (task4, 20), (task5, 100)]**

**last\_execution\_times = [0] \* len(tasks)**

**# Run cyclic executive**

**while True:**

**for i in range(len(tasks)):**

**if time.ticks\_diff(time.ticks\_ms(), last\_execution\_times[i]) >= tasks[i][1]:**

**tasks[i][0]()**

**last\_execution\_times[i] = time.ticks\_ms()**

**import machine**

**import time**

**# Define pin numbers for LED and analogue input**

**led\_pin = 2**

**analog\_pin = 36**

**# Initialize LED pin as output**

**led = machine.Pin(led\_pin, machine.Pin.OUT)**

**# Initialize analogue input pin**

**analog\_in = machine.ADC(analog\_pin)**

**# Define filter parameters for Task 4**

**FILTER\_SIZE = 4**

**analog\_filter = [0]\*FILTER\_SIZE**

**# Define variables for Task 2 and 3**

**freq1 = 0**

**freq2 = 0**

**# Define functions for each task**

**def task1():**

**# Output digital signal pattern**

**led.on()**

**time.sleep\_us(200)**

**led.off()**

**time.sleep\_us(50)**

**led.on()**

**time.sleep\_us(200)**

**led.off()**

**time.sleep\_us(50)**

**led.off()**

**time.sleep\_us(50)**

**led.on()**

**time.sleep\_us(200)**

**led.on()**

**time.sleep\_us(30)**

**led.off()**

**time.sleep\_us(50)**

**led.on()**

**def task2():**

**# Measure frequency of square wave signal on pin X**

**# Compute frequency and store in freq1 variable**

**# Frequency measurement code here**

**freq1 = 0**

**def task3():**

**# Measure frequency of square wave signal on pin Y**

**# Compute frequency and store in freq2 variable**

**# Frequency measurement code here**

**freq2 = 0**

**def task4():**

**# Read analogue input and compute filtered value**

**analog\_filter.pop(0)**

**analog\_filter.append(analog\_in.read())**

**filtered\_value = sum(analog\_filter) / FILTER\_SIZE**

**if filtered\_value > (3.3 / 2):**

**led.on()**

**else:**

**led.off()**

**def task5():**

**# Log freq1 and freq2 to serial port**

**print("%d,%d" % (freq1, freq2))**

**# Define task timings**

**task1\_period = 4000 # microseconds**

**task2\_period = 20000 # microseconds**

**task3\_period = 8000 # microseconds**

**task4\_period = 20000 # microseconds**

**task5\_period = 100000 # microseconds**

**# Define last execution times for each task**

**last\_task1\_time = 0**

**last\_task2\_time = 0**

**last\_task3\_time = 0**

**last\_task4\_time = 0**

**last\_task5\_time = 0**

**# Main loop**

**while True:**

**# Get current time**

**current\_time = time.ticks\_us()**

**# Execute Task 1 if it's time**

**if (current\_time - last\_task1\_time) >= task1\_period:**

**task1()**

**last\_task1\_time = current\_time**

**# Execute Task 2 if it's time**

**if (current\_time - last\_task2\_time) >= task2\_period:**

**task2()**

**last\_task2\_time = current\_time**

**# Execute Task 3 if it's time**

**if (current\_time - last\_task3\_time) >= task3\_period:**

**task3()**

**last\_task3\_time = current\_time**

**# Execute Task 4 if it's time**

**if (current\_time - last\_task4\_time) >= task4\_period:**

**task4()**

**last\_task4\_time = current\_time**

**# Execute Task 5 if it's time**

**if (current\_time - last\_task5\_time) >= task5\_period:**

**task5()**

**last\_task5\_time=current\_time**

**import machine**

**import utime**

**# Define constants**

**PIN\_LED = 5**

**PIN\_ANALOG\_IN = 34**

**# Define tasks**

**def task\_1():**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(1) # HIGH for 200us**

**utime.sleep\_us(200)**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(0) # LOW for 50us**

**utime.sleep\_us(50)**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(1) # HIGH for 50us**

**utime.sleep\_us(50)**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(0) # LOW for 50us**

**utime.sleep\_us(50)**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(0) # LOW for 50us**

**utime.sleep\_us(50)**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(1) # HIGH for 30us**

**utime.sleep\_us(30)**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(0) # LOW for 50us**

**utime.sleep\_us(50)**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(1) # HIGH for 200us**

**utime.sleep\_us(200)**

**def task\_2():**

**freq = machine.freq\_counter(machine.Pin(0))**

**print("%d," % freq, end='')**

**def task\_3():**

**freq = machine.freq\_counter(machine.Pin(2))**

**print("%d" % freq)**

**def task\_4():**

**sum\_analog\_in = 0**

**for i in range(4):**

**sum\_analog\_in += machine.ADC(PIN\_ANALOG\_IN).read()**

**utime.sleep\_ms(5)**

**average\_analog\_in = sum\_analog\_in / 4**

**if average\_analog\_in > 1650: # half of maximum range**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(1)**

**else:**

**machine.Pin(PIN\_LED, machine.Pin.OUT).value(0)**

**def task\_5():**

**task\_2()**

**task\_3()**

**# Define cyclic executive**

**def cyclic\_executive():**

**while True:**

**start\_time = utime.ticks\_ms()**

**task\_1()**

**if utime.ticks\_diff(utime.ticks\_ms(), start\_time) >= 4:**

**continue**

**task\_2()**

**if utime.ticks\_diff(utime.ticks\_ms(), start\_time) >= 20:**

**continue**

**task\_3()**

**if utime.ticks\_diff(utime.ticks\_ms(), start\_time) >= 28:**

**continue**

**task\_4()**

**if utime.ticks\_diff(utime.ticks\_ms(), start\_time) >= 100:**

**continue**

**task\_5()**

**# Start cyclic executive**

**cyclic\_executive()**