

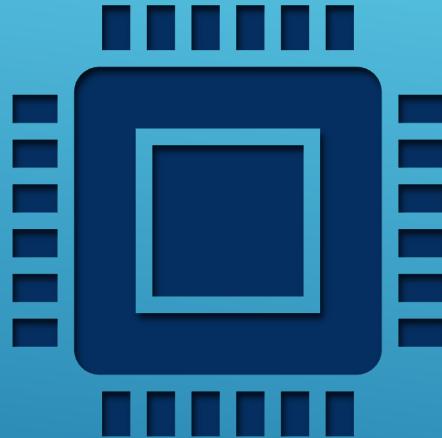
2024 Semester 1

# EE 01205479

# INTERNET OF THINGS FOR

# ELECTRICAL ENGINEERING

Dr. Varodom Toochinda  
RDiPT, ME, Kasetsart University



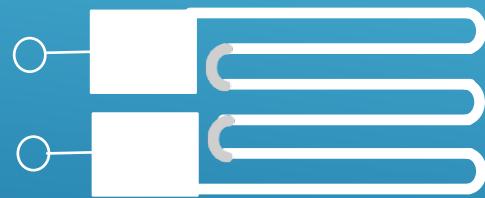
## COURSE ESSENTIALS

- ▶ Emphasize embedded systems and Internet of Things (IoT)
- ▶ Basic electronic/electrical circuits or analog interfacing
- ▶ Hardware (ESP8266/ESP32) and software (Arduino IDE, Scilab, Python+Jupyter Notebook, Micropython)
- ▶ IoT cloud platform (NETPIE)
- ▶ Continuous and discrete LTI systems
- ▶ Digital signal processing
- ▶ Real-time multitasking system design
- ▶ Advanced Topics

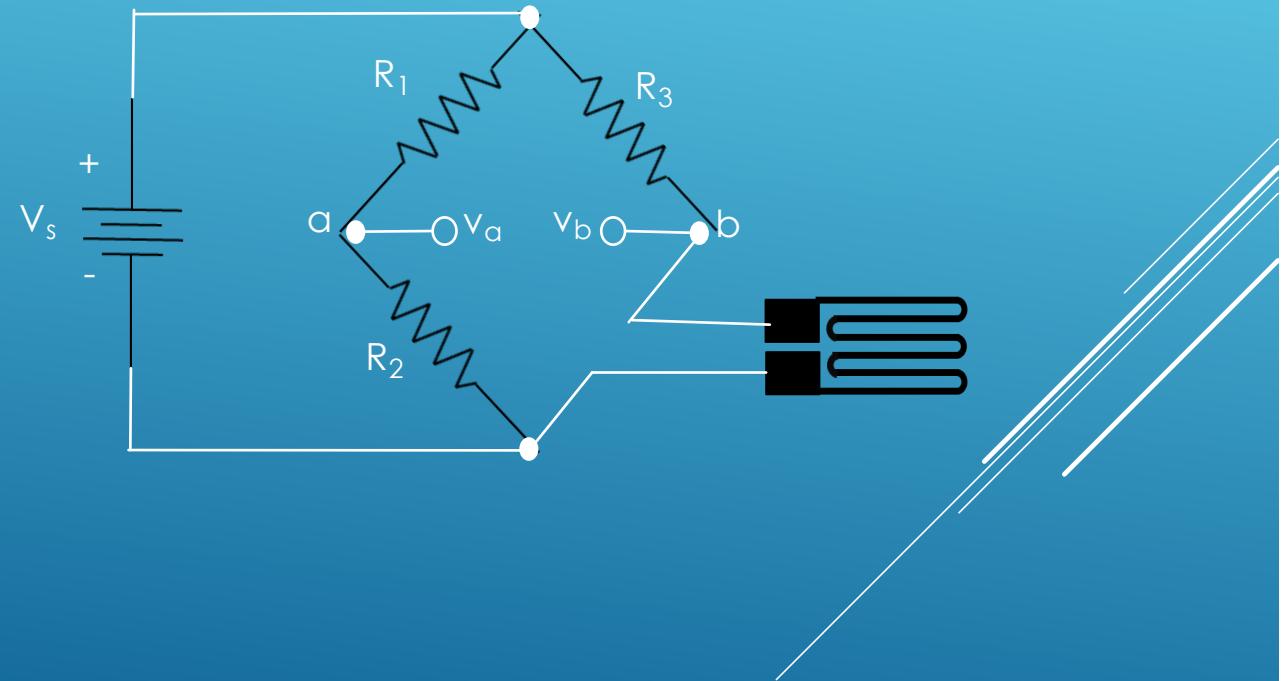
# BASIC CIRCUITS & SIGNAL CONDITIONING

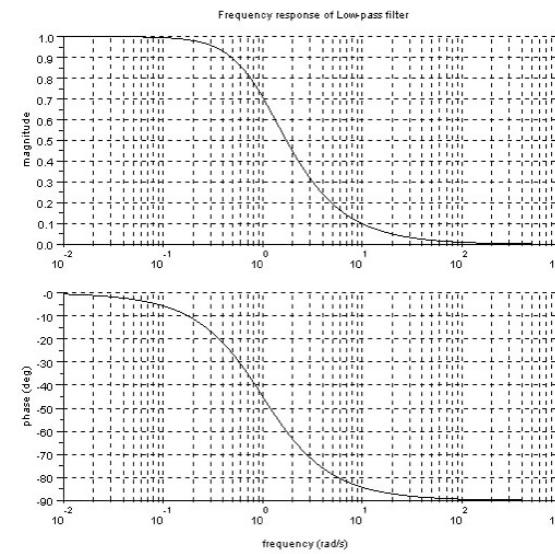
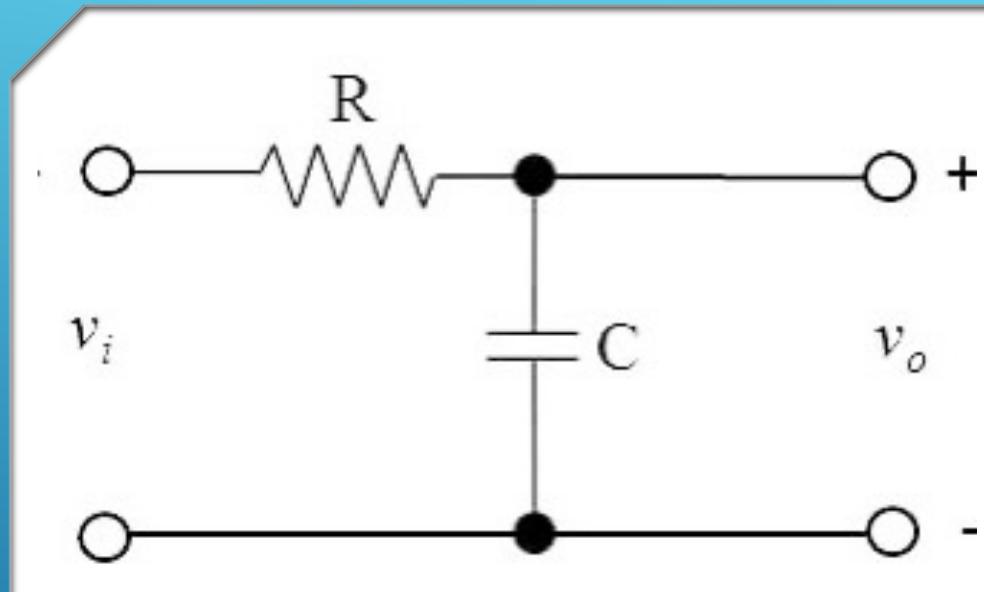
In order to develop a successful IoT device, we need to know how to design hardware and interfacing circuit.

# DC CIRCUIT EXAMPLE : RESISTIVE NETWORK

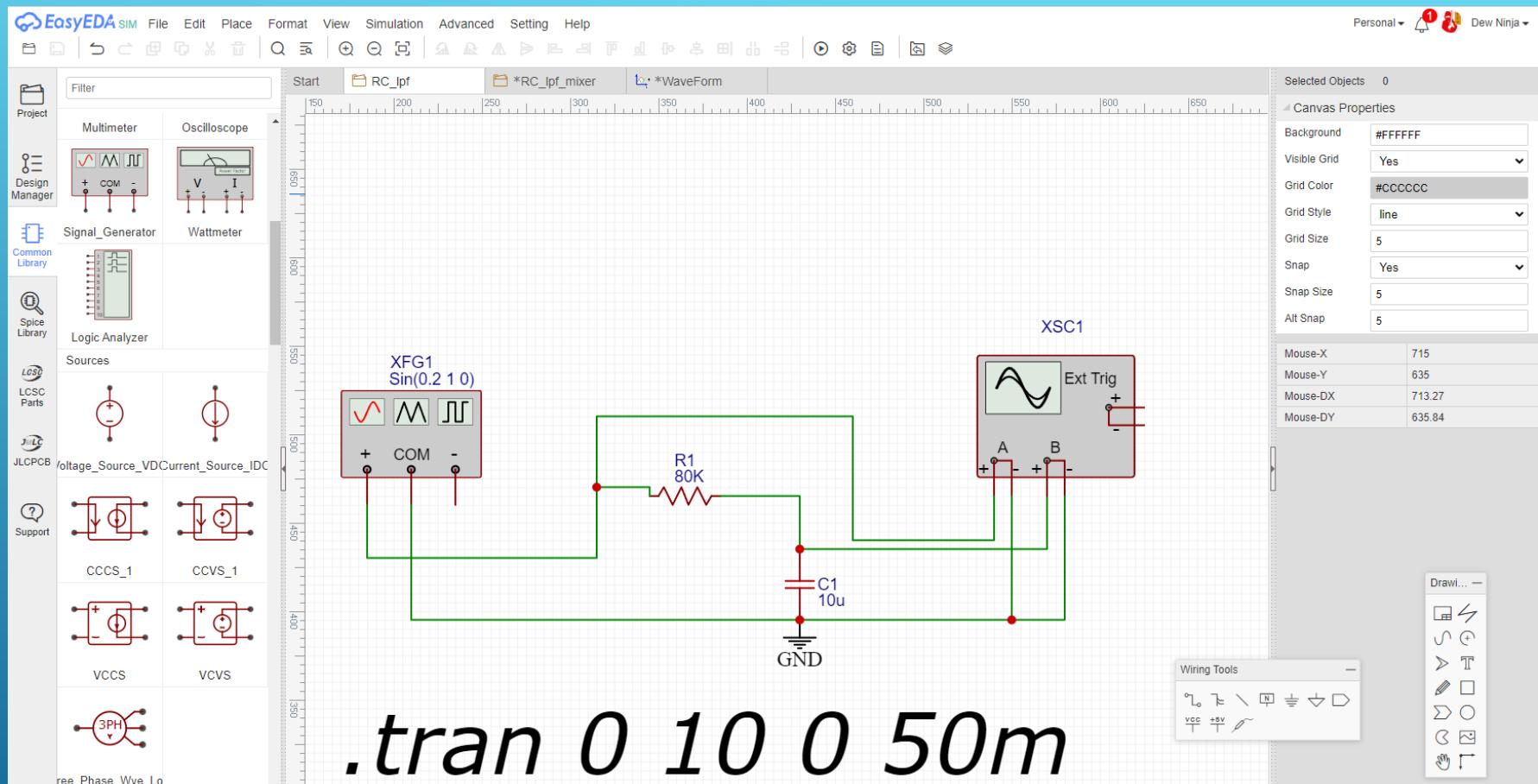


Stain gauge

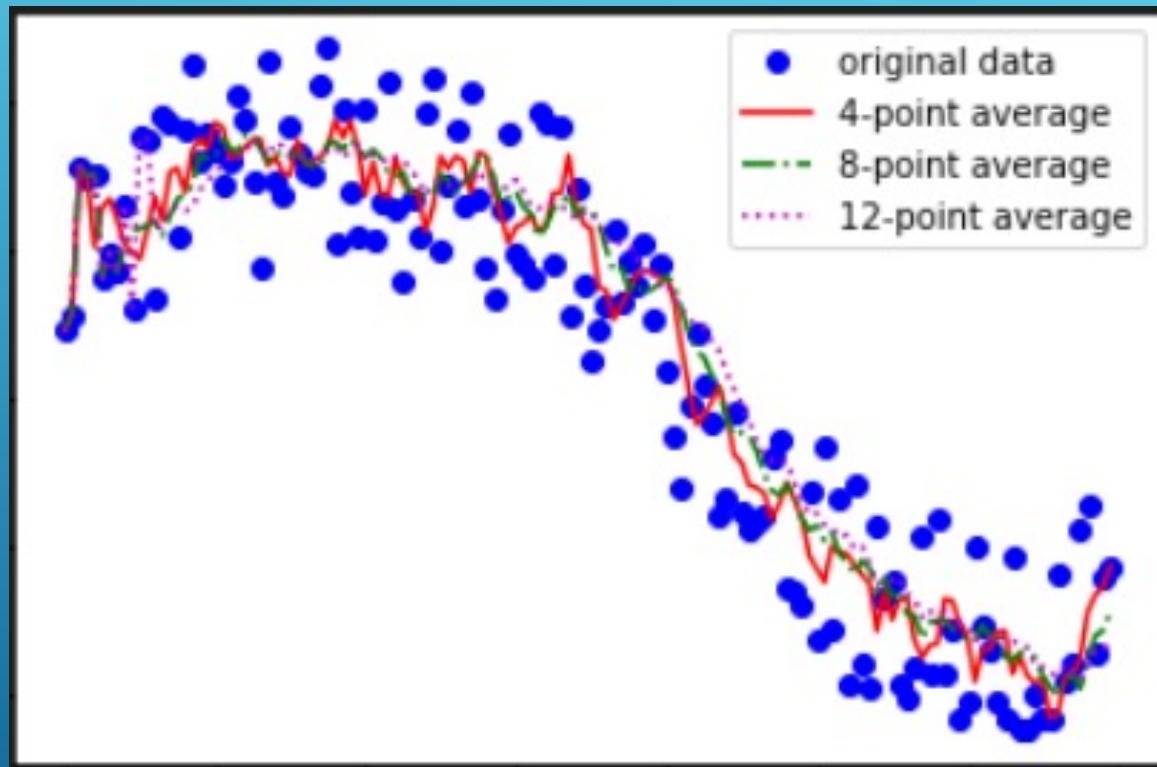




## AC CIRCUIT EXAMPLE : PASSIVE LPF

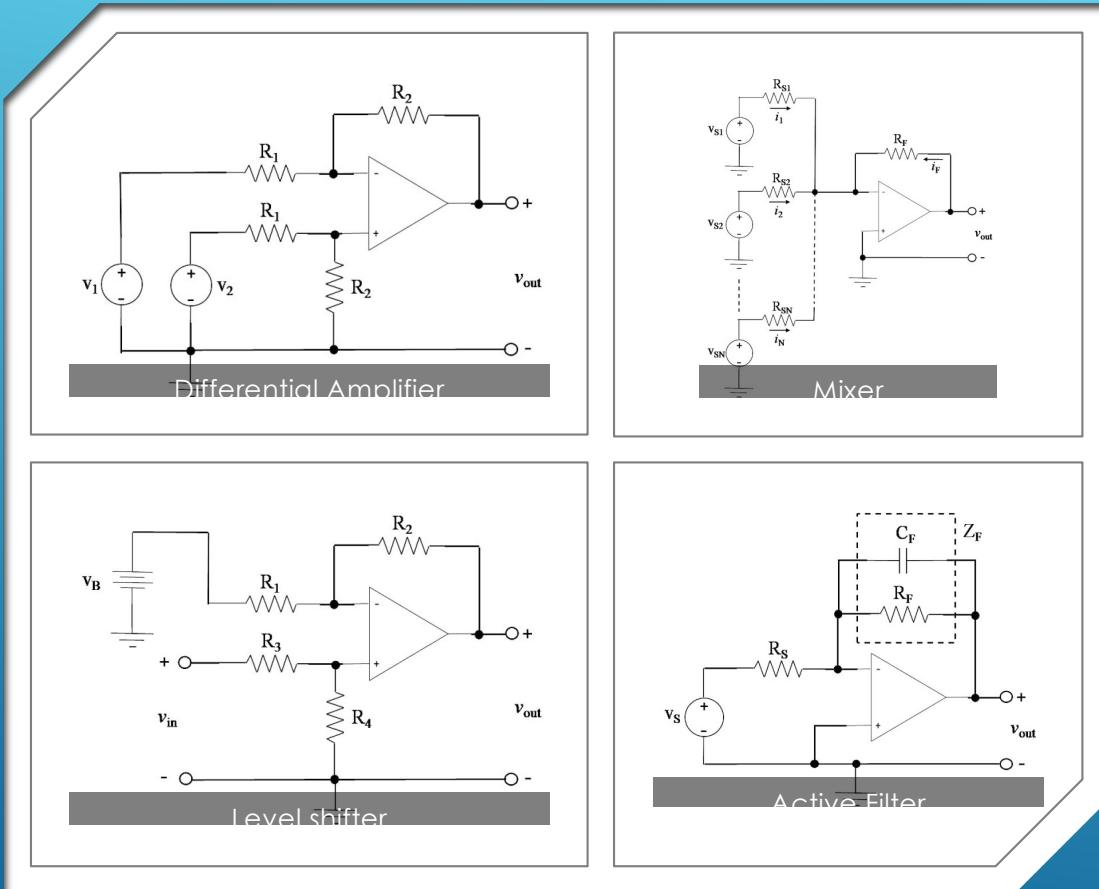


# LPF SIMULATION ON EASYEDA

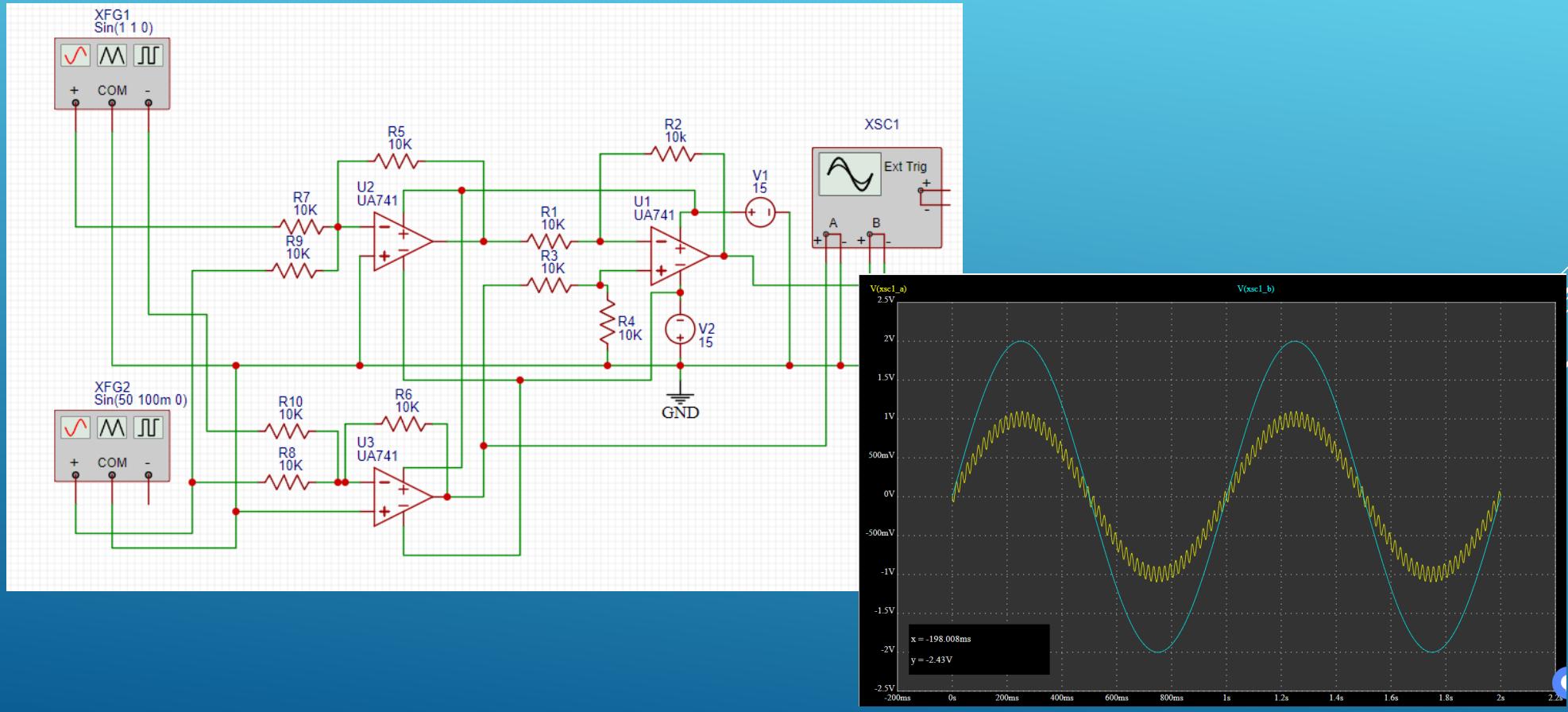


# NOISE FILTERING BY AVERAGING

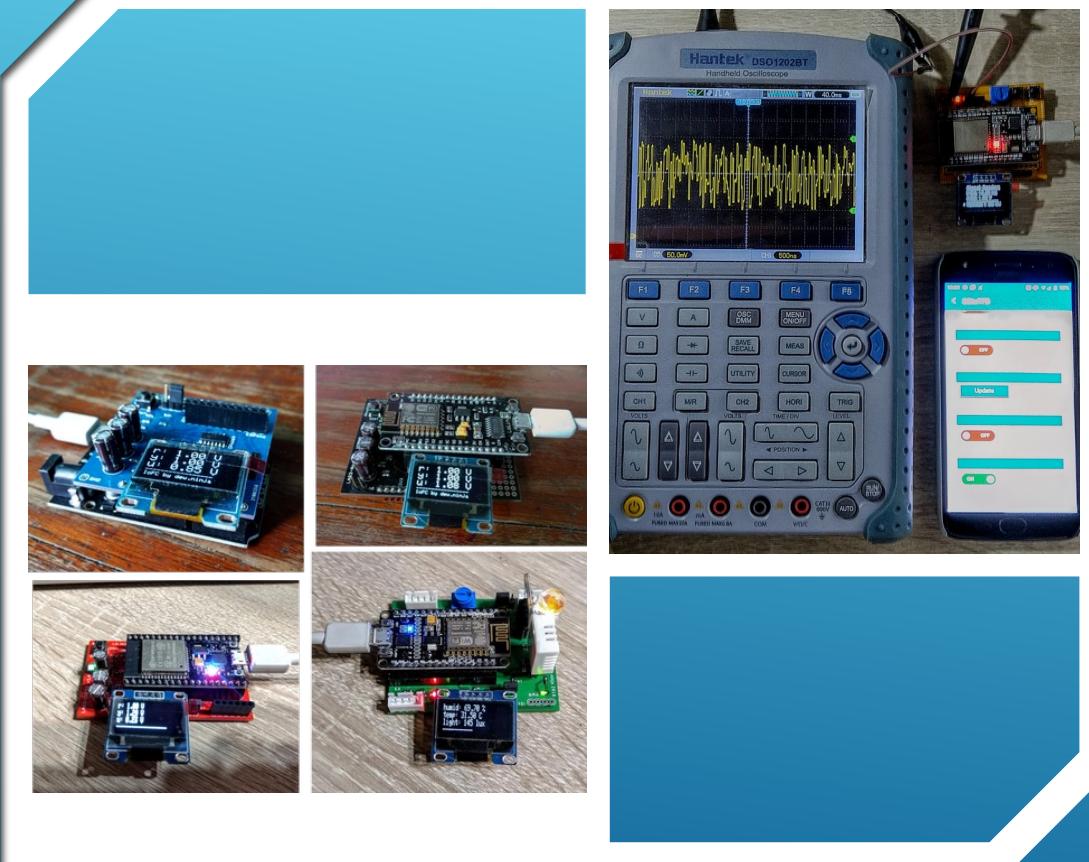
# OPERATIONAL AMPLIFIER EXAMPLE

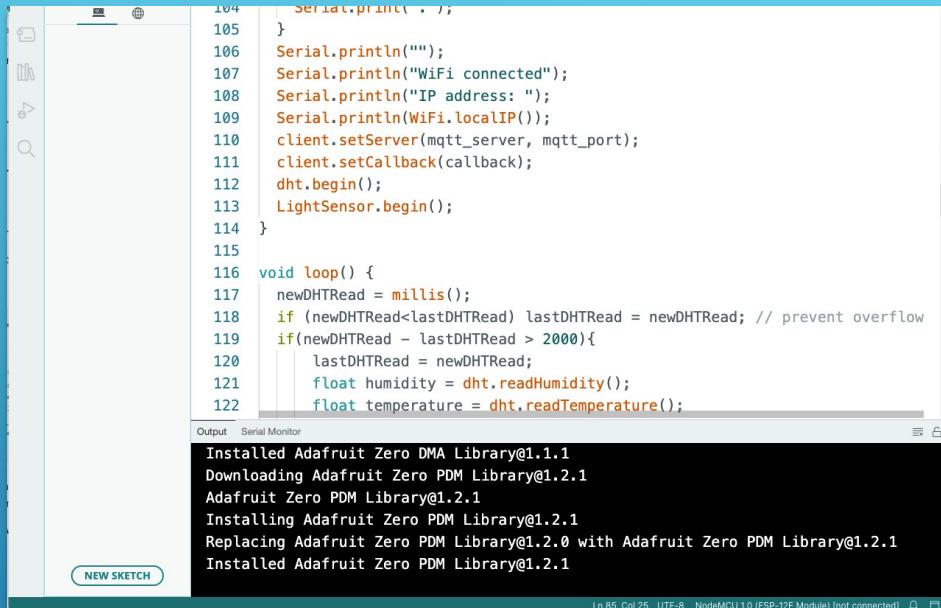


# Differential amp simulation



# IOT-BASED HARDWARE





```
104     Serial.println(" ");
105 }
106 Serial.println("");
107 Serial.println("WiFi connected");
108 Serial.println("IP address: ");
109 Serial.println(WiFi.localIP());
110 client.setServer(mqtt_server, mqtt_port);
111 client.setCallback(callback);
112 dht.begin();
113 LightSensor.begin();
114 }

115 void loop() {
116     newDHTRead = millis();
117     if (newDHTRead<lastDHTRead) lastDHTRead = newDHTRead; // prevent overflow
118     if(newDHTRead - lastDHTRead > 2000){
119         lastDHTRead = newDHTRead;
120         float humidity = dht.readHumidity();
121         float temperature = dht.readTemperature();
```

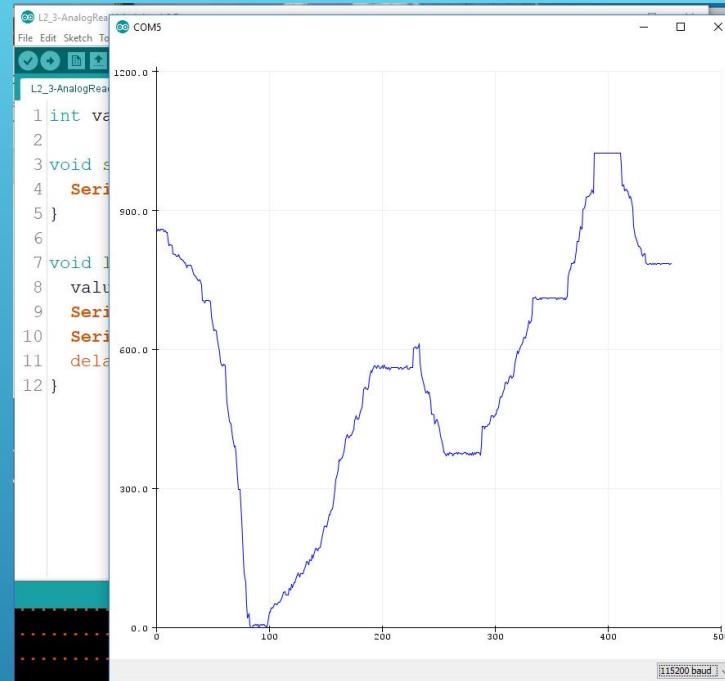
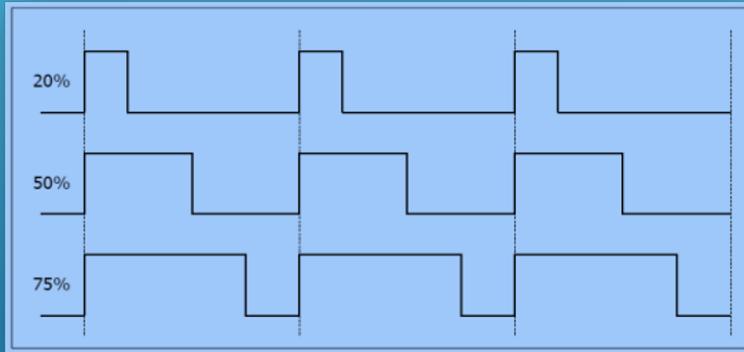
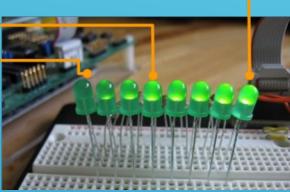
Output: Serial Monitor

```
Installed Adafruit Zero DMA Library@1.1.1
Downloading Adafruit Zero PDM Library@1.2.1
Adafruit Zero PDM Library@1.2.1
Installing Adafruit Zero PDM Library@1.2.1
Replacing Adafruit Zero PDM Library@1.2.0 with Adafruit Zero PDM Library@1.2.1
Installed Adafruit Zero PDM Library@1.2.1
```

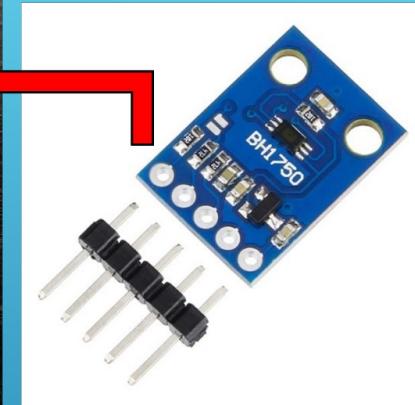
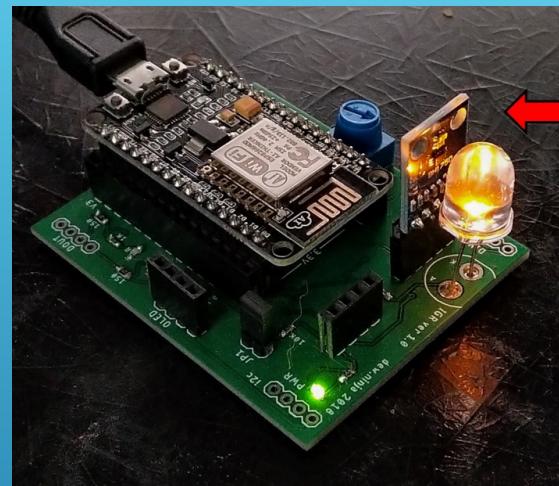
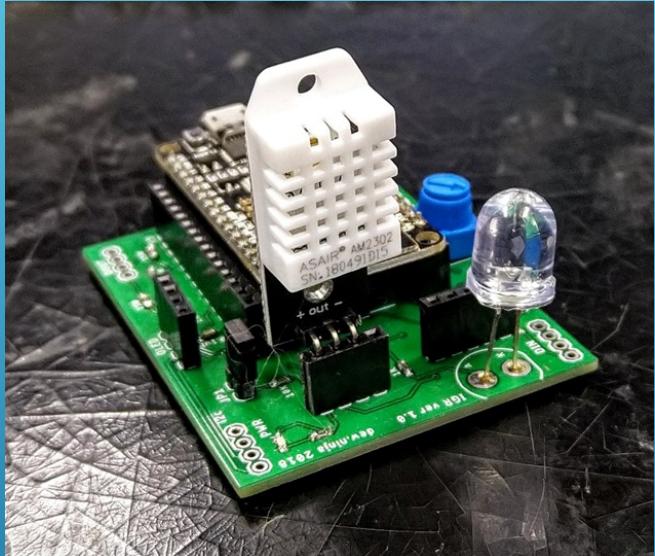


# C++ PROGRAMMING ON ESP8266/ESP32

```
analogWrite(D5,1023);  
analogWrite(D5,512);  
analogWrite(D5,0);
```

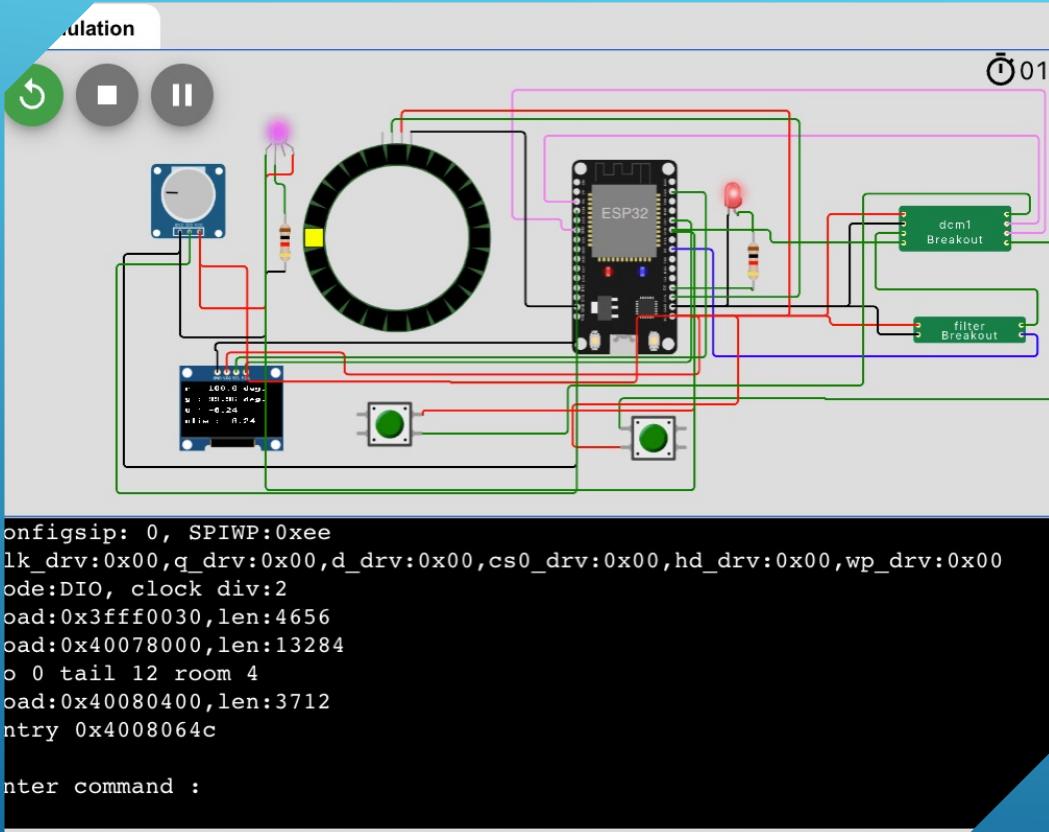


## BASIC I/O'S

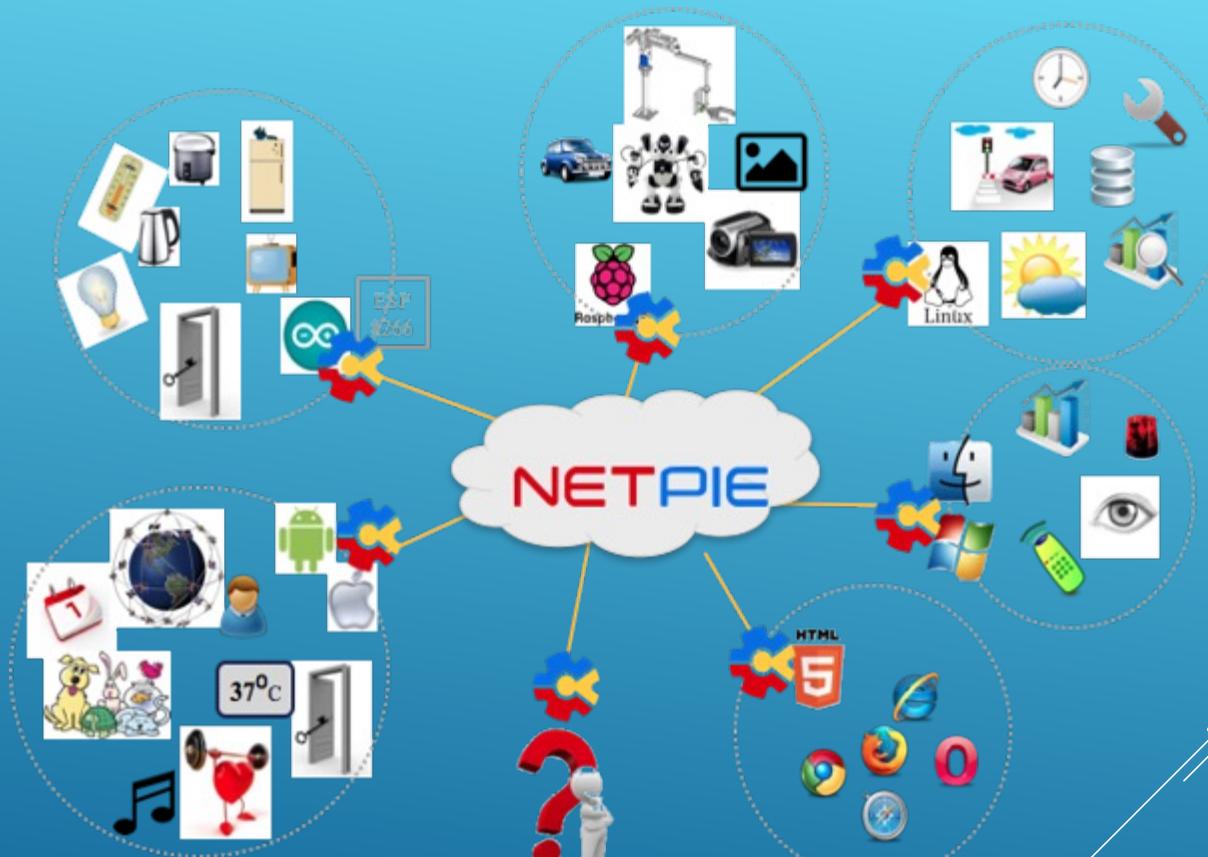


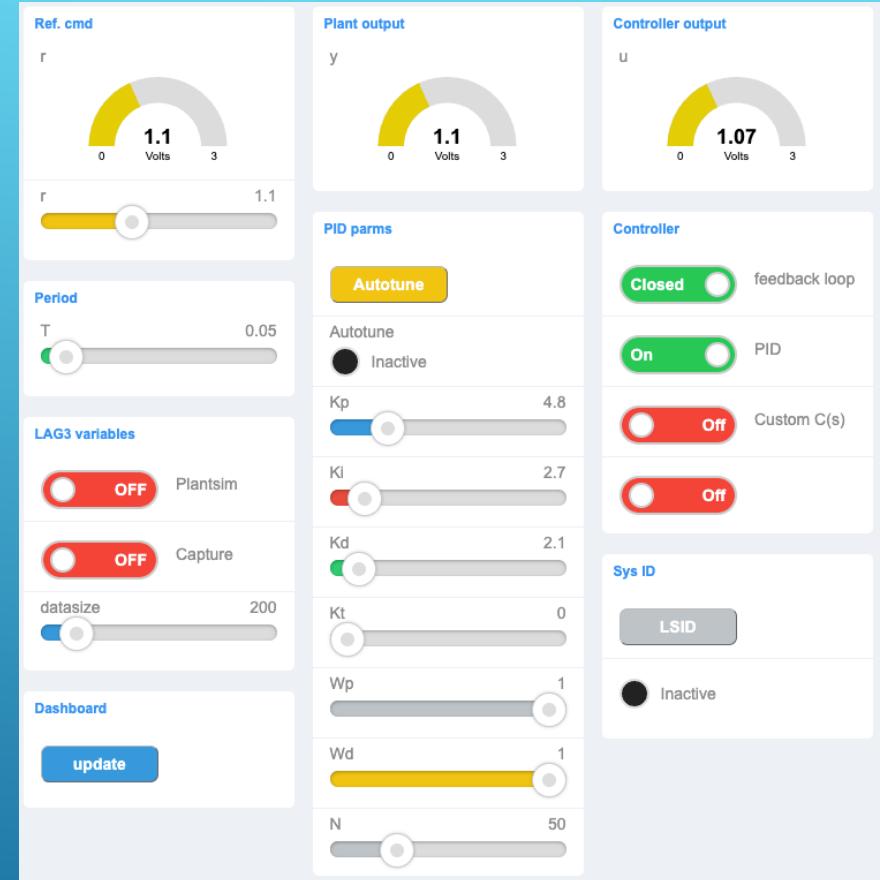
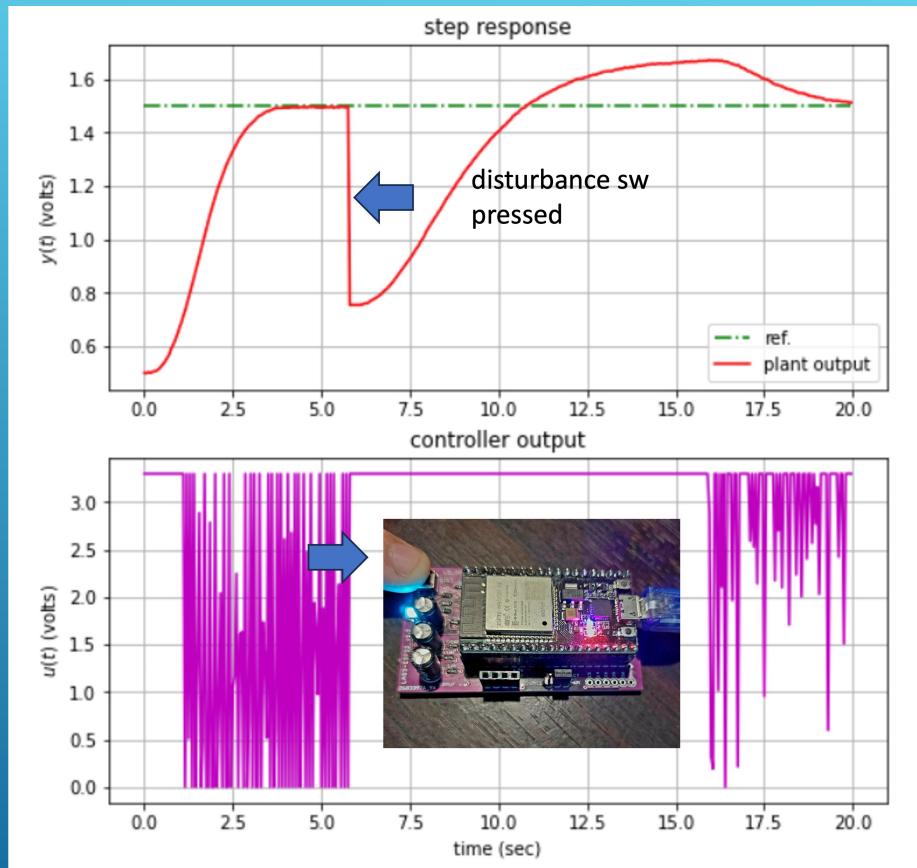
## SENSORS INTERFACE

# EMBEDDED SYSTEM SIMULATION



# NETPIE IOT





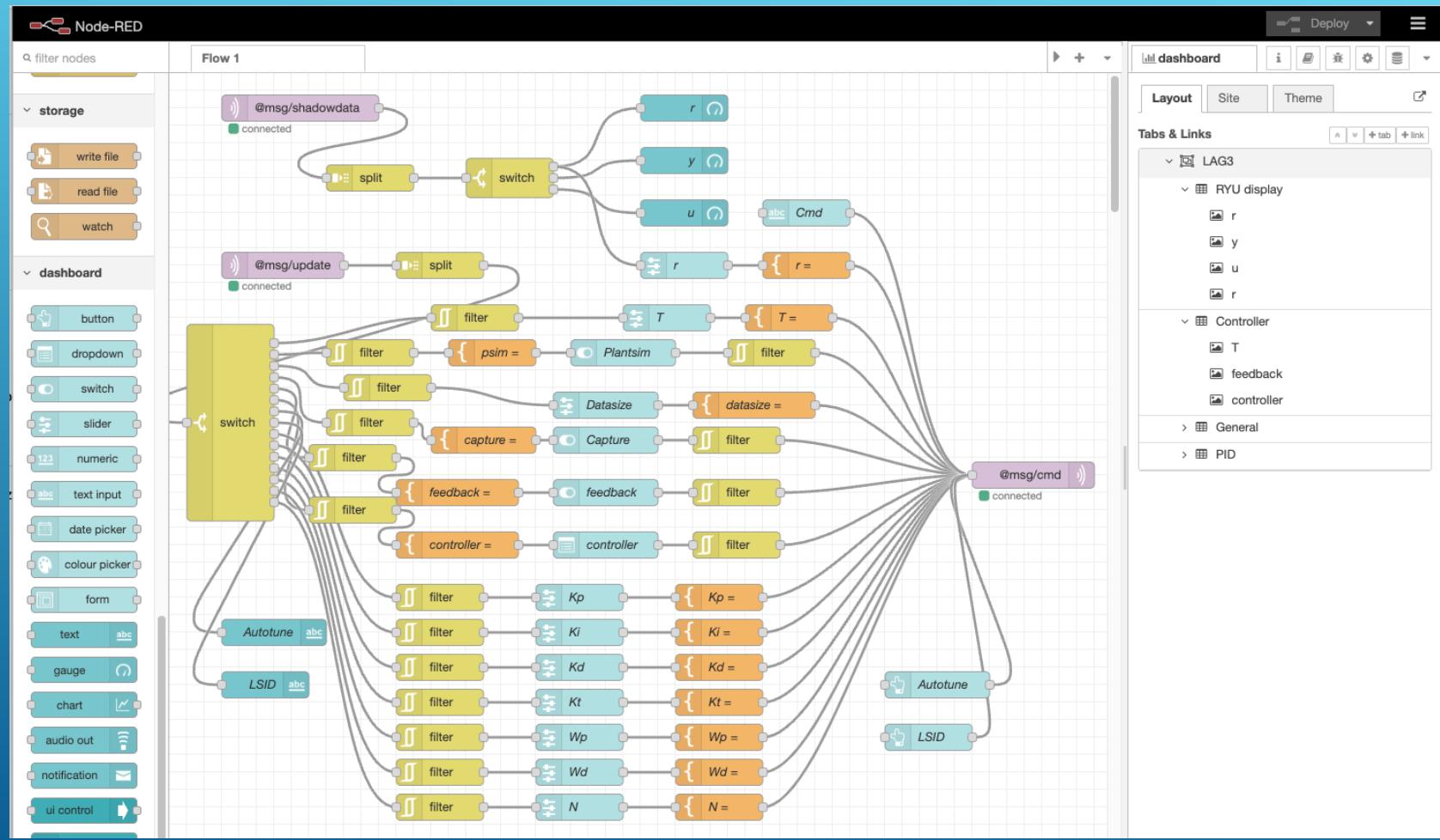
# IoT for industrial applications

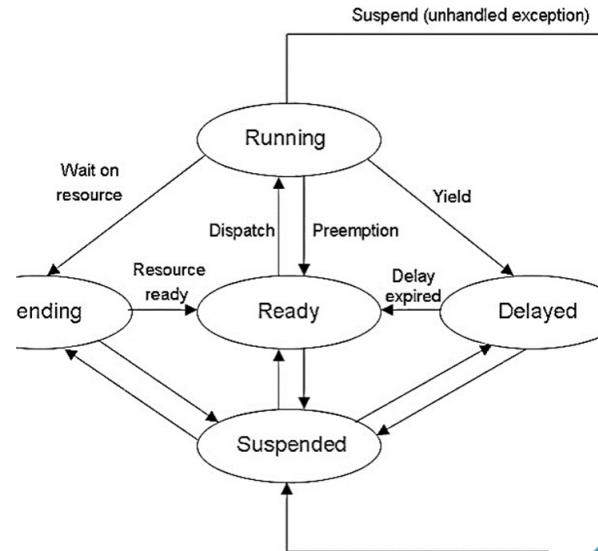
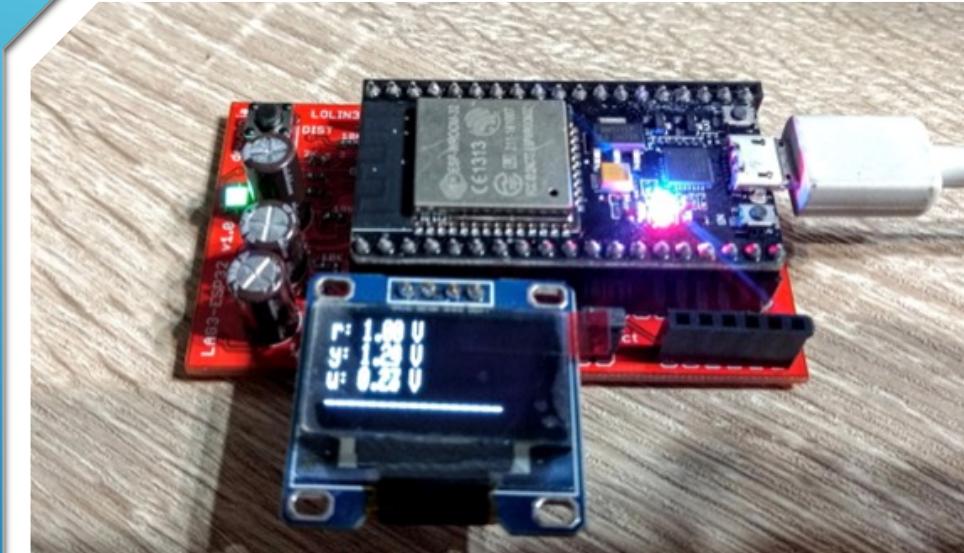
# MICROPYTHON

The image displays a Mac desktop environment with three open windows:

- Jupyter Notebook:** Titled "olresp\_compare2" (autosaved). It contains Python code for plotting the "Open loop response". The plot shows two curves: "Ref. cmd r(t)" (black dashed line) and "Plant output (PC)" (blue solid line) and "Plant output (ESP32)" (red solid line). The x-axis is labeled "time (s)" and ranges from 0 to 10. The y-axis is labeled "Volts" and ranges from 0.0 to 1.0.
- Hardware:** An ESP32 Dev Board is shown connected to a computer via a USB cable. The board has various components like a WiFi module, a microSD card slot, and a small LCD screen.
- Terminal:** Titled "netpie\_cmdtr" with the command "python3 lag3\_ol\_net.py" running. It shows a script for handling network commands. The script includes logic for capturing data and updating the freeboard. It also handles commands like "datasize" and "update\_freeboard".

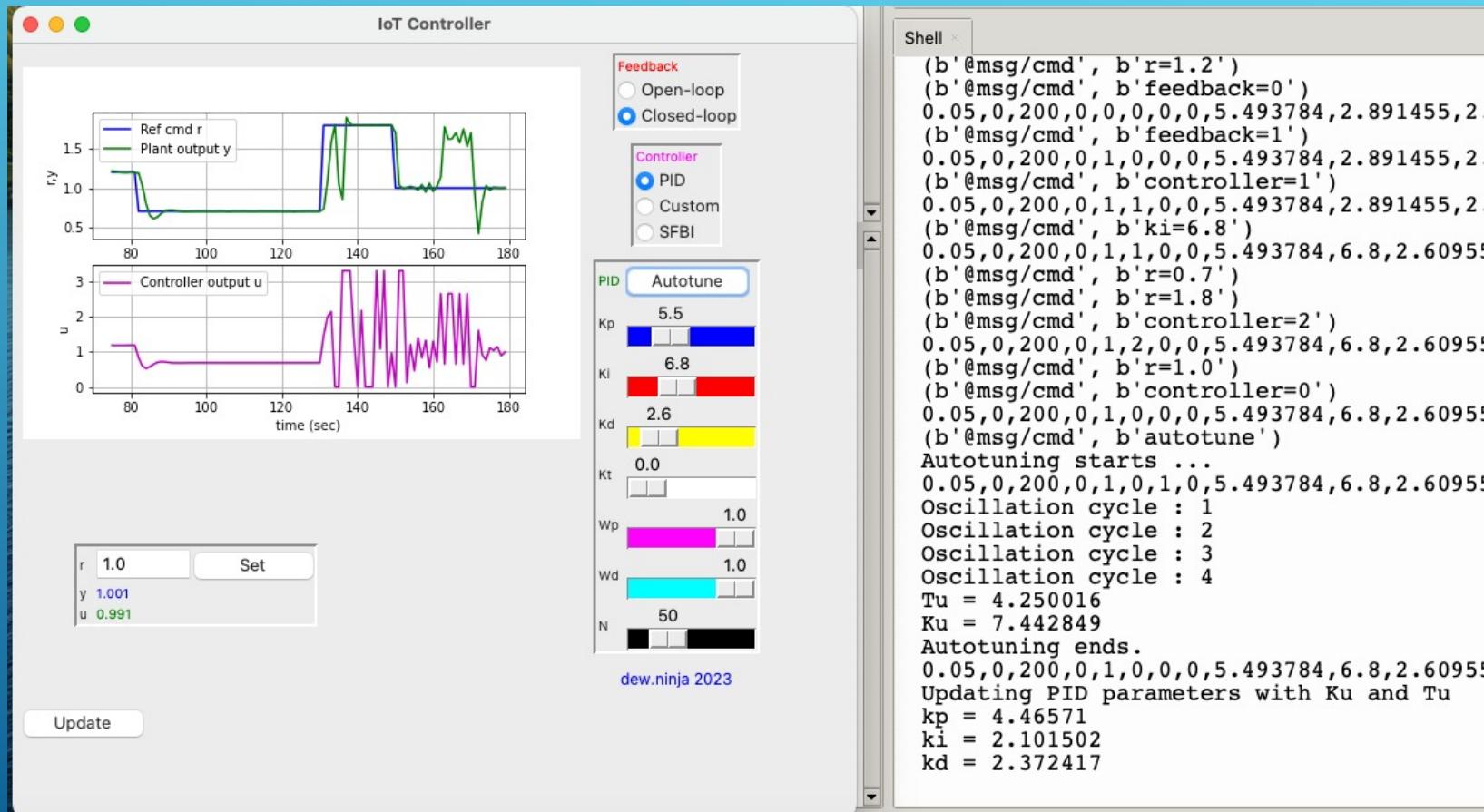
# NODE-RED





# REAL-TIME MULTITASKING WITH FREERTOS

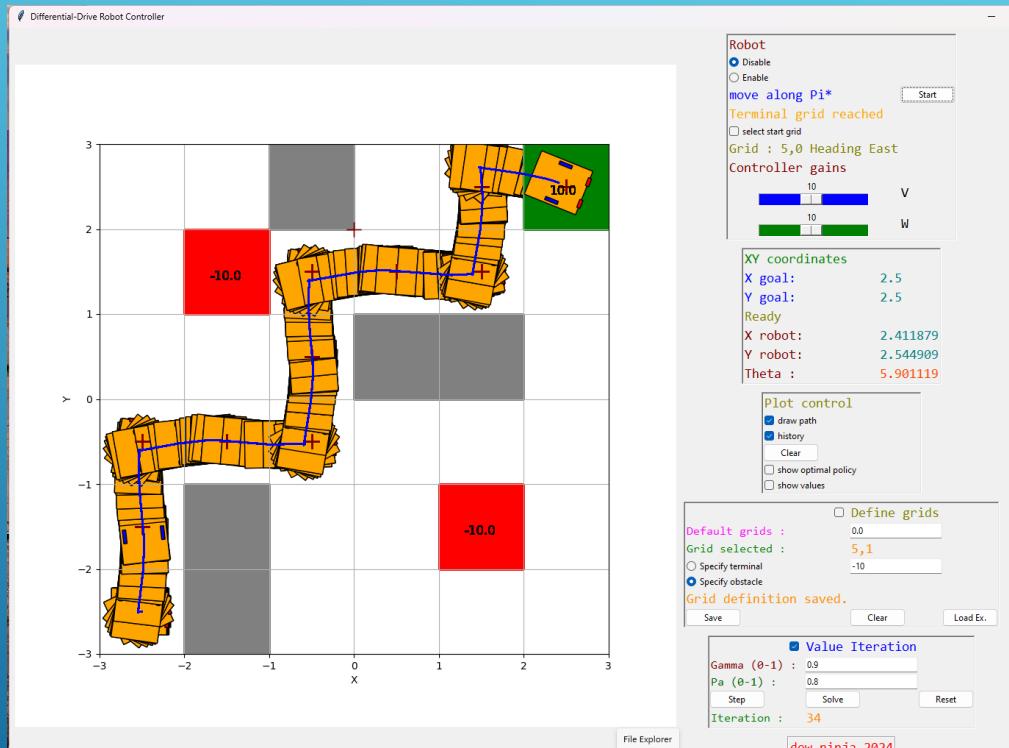
# OOP (OBJECT-ORIENTED PROGRAMMING)

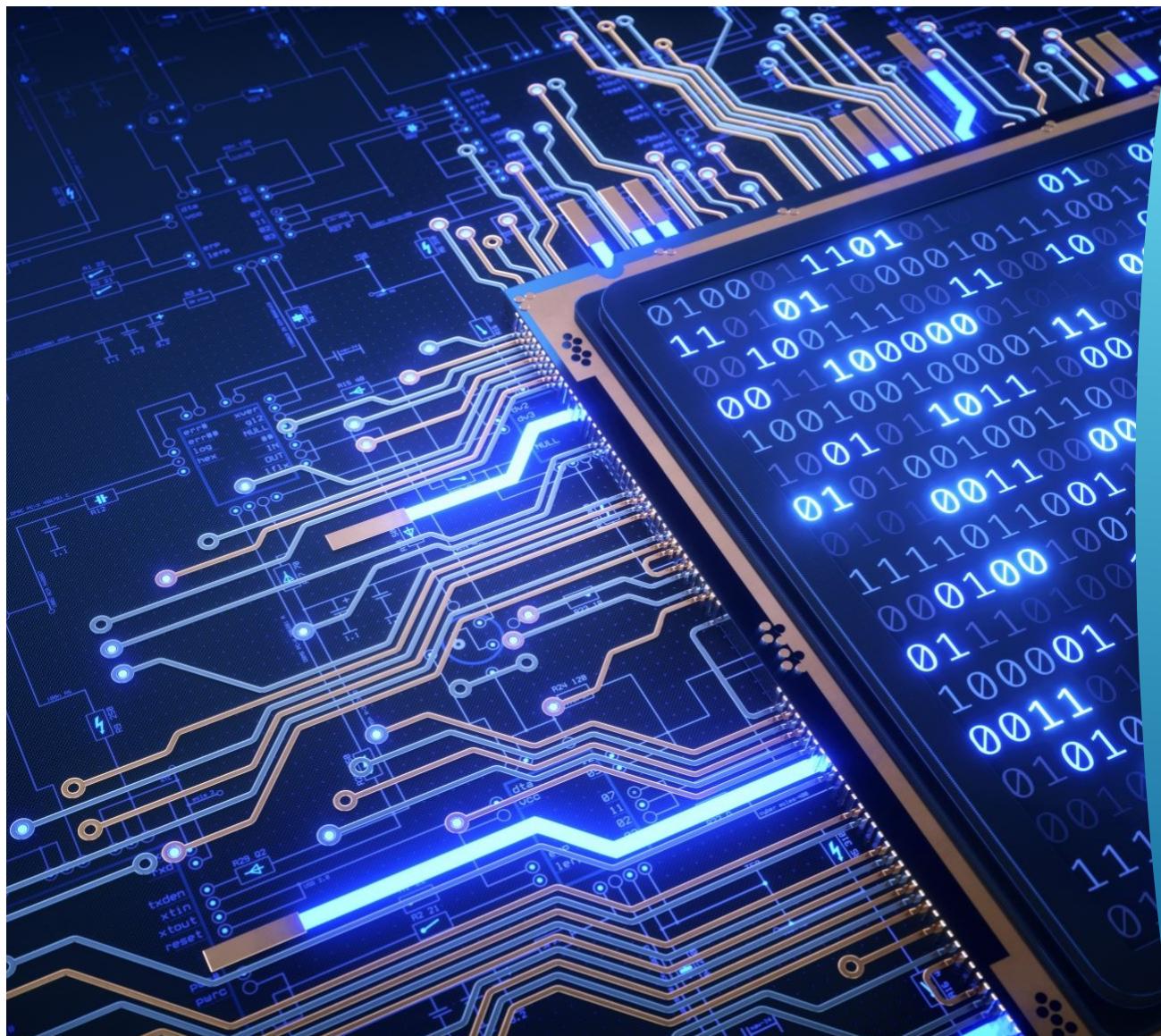


# ADVANCED TOPICS & CASE STUDY

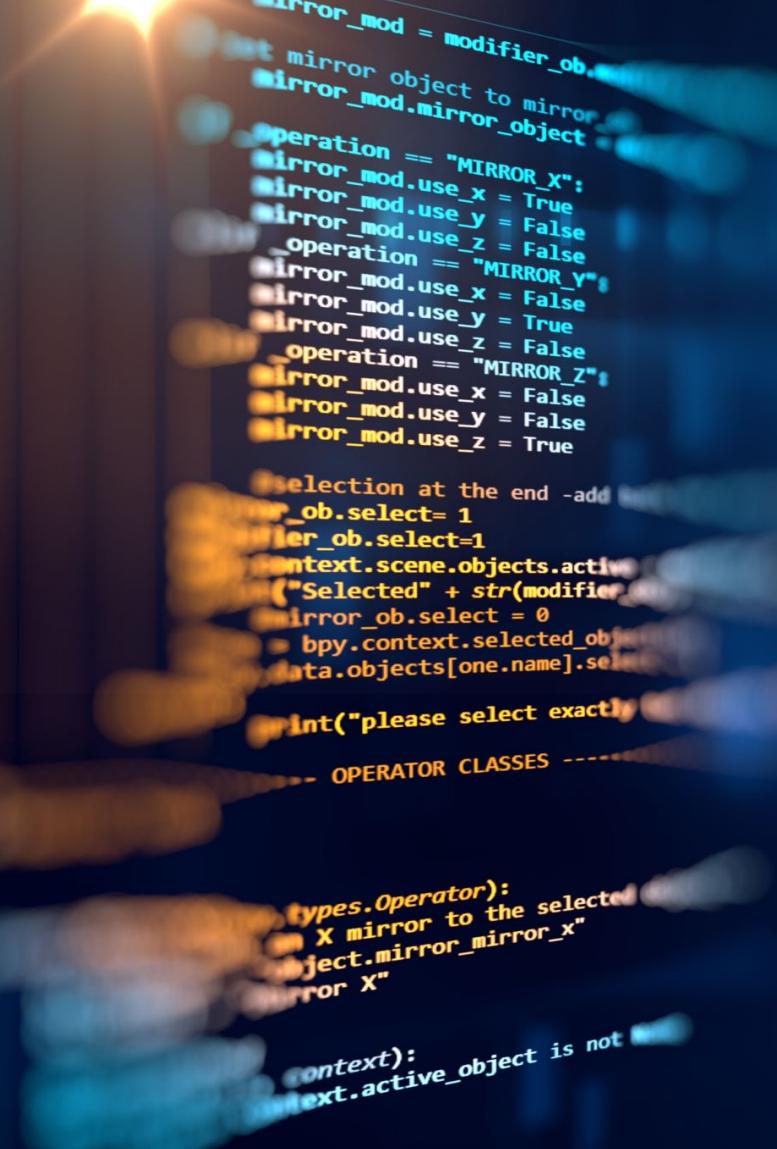
\* as time permits

# INTEGRATE IOT WITH REINFORCEMENT LEARNING





# COMPUTER PROGRAMMING



# PROGRAMMING LANGUAGES USED IN THIS COURSE

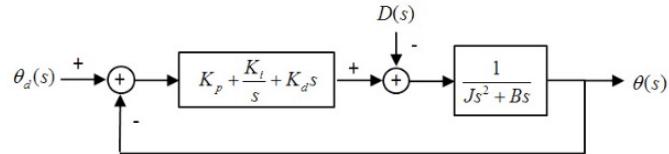
- ▶ C (and some C++) : for Arduino platform
- ▶ Python : for Jupyter notebook and ESP32 (micropython)
- ▶ JavaScript : some IoT programming

# SW (OPEN-SOURCE OR FREE FOR BASIC USAGE)

- ▶ Arduino IDE and libraries
- ▶ NETPIE 2020 : <https://netpie.io/>
- ▶ micropython and libraries
- ▶ Thonny
- ▶ Jupyter notebook (colab or local)

# PYTHON PROGRAMMING ON JUPYTER NOTEBOOK

**Example 2:** To experiment with the tracking and disturbance attenuation performance of PID control, we setup a feedback diagram in Figure 2.



**Figure 2 PID feedback with input disturbance**

Such diagram is quite easy to construct using simulation engine like Simulink in MATLAB, or Xcos in Scilab. It is more tricky using time-domain simulation function in Python Control Systems library.

First, it can be derived that the transfer function from  $D(s)$  to  $\theta(s)$  is  $P(s)S(s)$ . So the disturbance response is the output from this transfer function. Suppose that The step disturbance at the plant input is active at half of time vector, with step value of 80. Adding the outputs together yields the step response with step disturbance input.

```
In [4]:  
sys1 = ctrl.feedback(L) # system for step command  
S = 1/(1+L)  
sys2 = P*S  
  
tvec = np.arange(0,20,0.01) # adjust time vector if necessary  
r = np.ones(tvec.shape)  
d = 80*np.ones(tvec.shape)  
d[int(len(tvec)/2)] = 0.0  
T1, y1 = ctrl.step_response(sys1, tvec)  
T2, y2 = ctrl.forced_response(sys2, tvec, d)  
y = y1 + y2
```

Note that in this example, we have the luxury of knowing the exact plant model. So it is possible to solve for a good set of PID gains. An approach used in [2] turns off the  $K_i$  gain first and solve for  $K_p$  and  $K_d$  that yields a critical-damped system at chosen natural frequency. Then  $K_i$  is adjusted to eliminate steady-state error.

To elaborate, it is shown in [2] that when  $K_i$  is set to 0, the resulting second order closed-loop characteristic polynomial is described by

$$s^2 + \frac{(B + K_d)}{J} s + \frac{K_p}{J} = s^2 + 2\zeta\omega s + \omega^2 \quad (7)$$

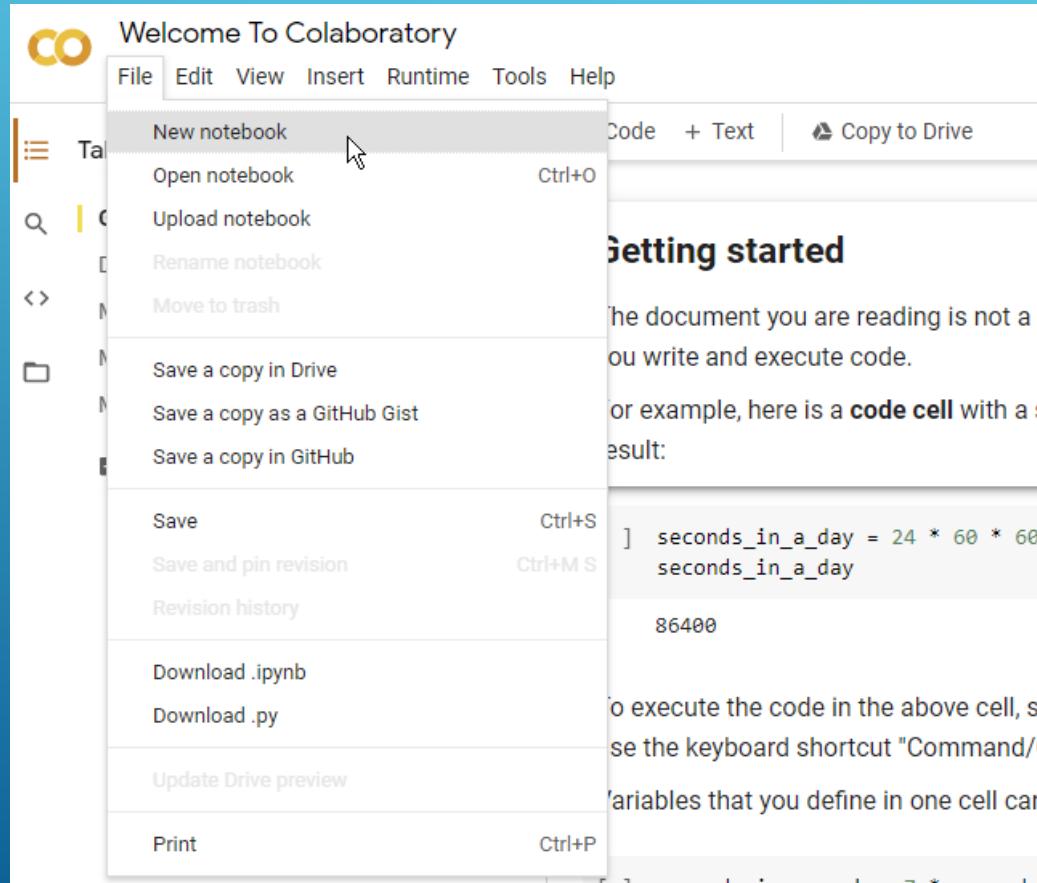
(7)

which gives

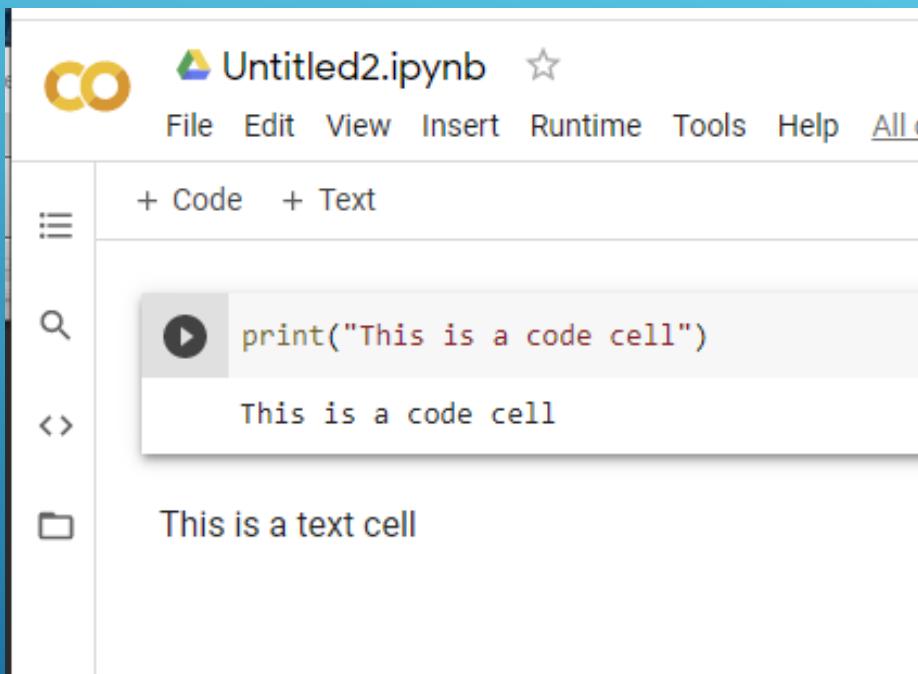
$$K_p = \omega^2 J, K_d = 2\zeta\omega J - B \quad (8)$$

(8)

# Begin Python Development in Colab



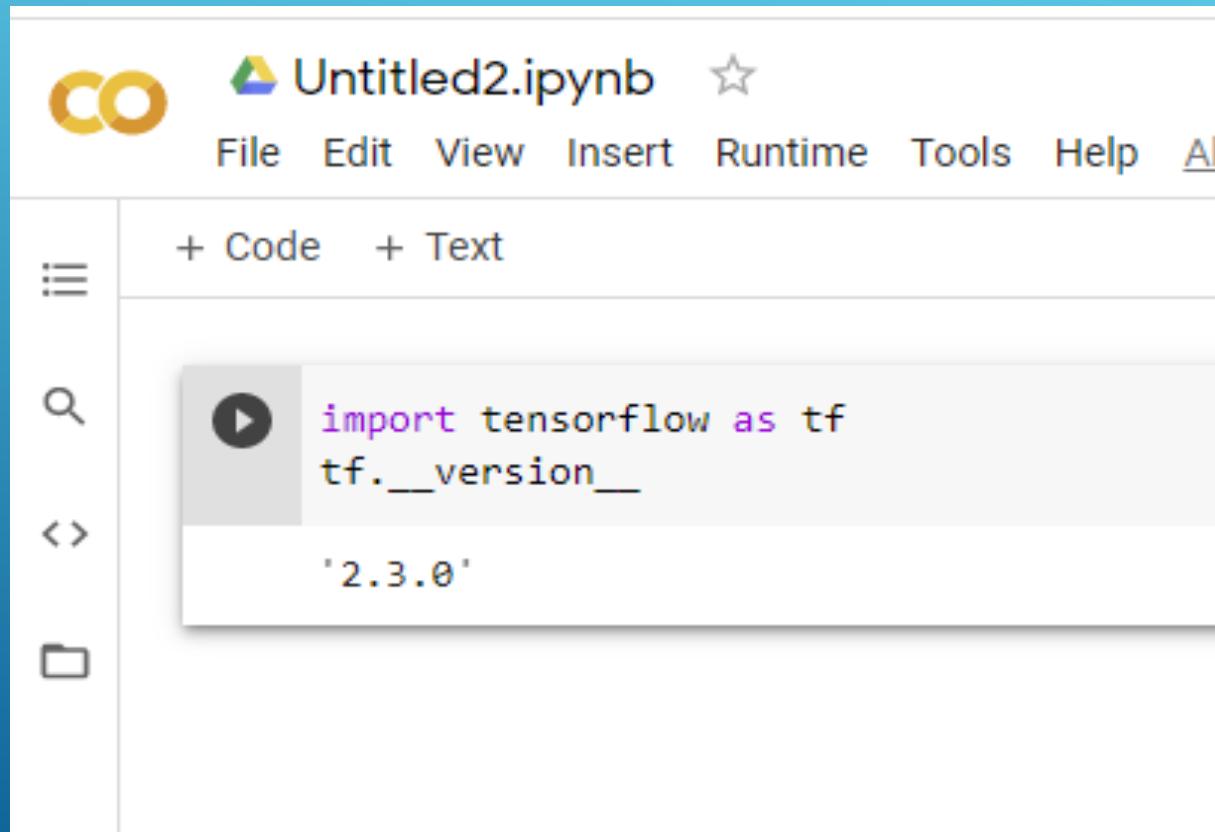
# 2 TYPES OF CELL IN NOTEBOOK



The screenshot shows a Jupyter Notebook interface with the title "Untitled2.ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and All. On the left, there's a sidebar with icons for file operations. The main area displays two cells:

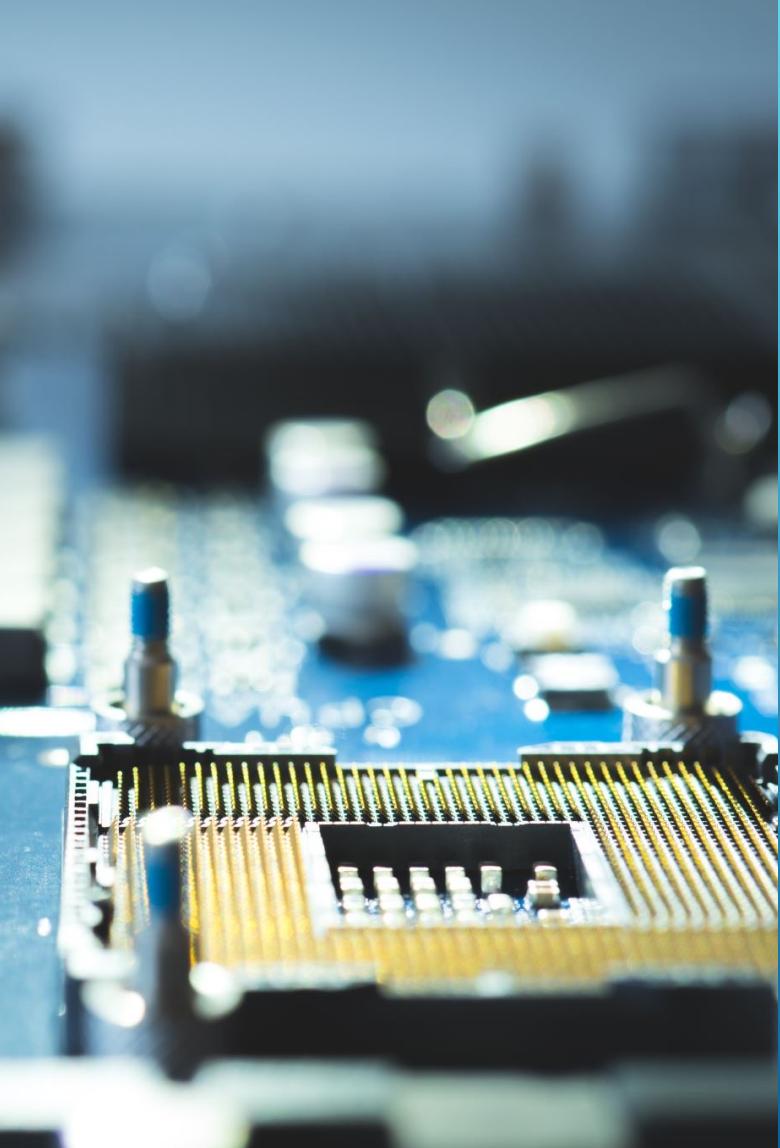
- A code cell containing the Python code: `print("This is a code cell")`. The output of this cell, "This is a code cell", is shown below it.
- A text cell containing the text "This is a text cell".

TensorFlow is already installed in Colab



The image shows a screenshot of the Google Colab interface. At the top, there's a navigation bar with a 'CO' logo, the file name 'Untitled2.ipynb', and icons for saving and starring. Below the bar are menu options: File, Edit, View, Insert, Runtime, Tools, Help, and All. On the left side, there's a sidebar with icons for file operations like new file, search, and refresh. The main workspace shows a code cell with a play button icon. Inside the cell, the following Python code is displayed:

```
import tensorflow as tf
tf.__version__
'2.3.0'
```



## TEXTS AND MATERIALS

- ▶ Texts
  - ▶ Handouts from instructor
- ▶ Online resources
  - ▶ [github.com/dewdotninja](https://github.com/dewdotninja)
  - ▶ NETPIE website <https://netpie.io/>
  - ▶ Arduino <https://www.arduino.cc/>