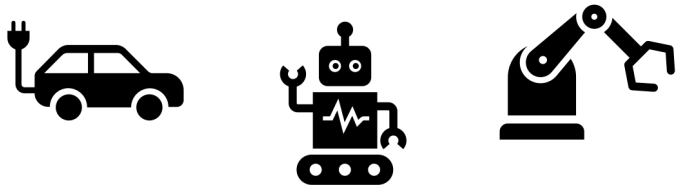
WASHINGTON STATE UNIVERSITY

CySER Workshop 2025

Faking Cyber-Physical Systems for Fun & Profit



Dr. Monowar Hasan

Assistant Professor, EECS@WSU

Mohammad Fakhruddin Babar

Research Assistant, EECS@WSU

Tamim Ahmed

Research Assistant, EECS@WSU

Tasnim Farhan Fatin

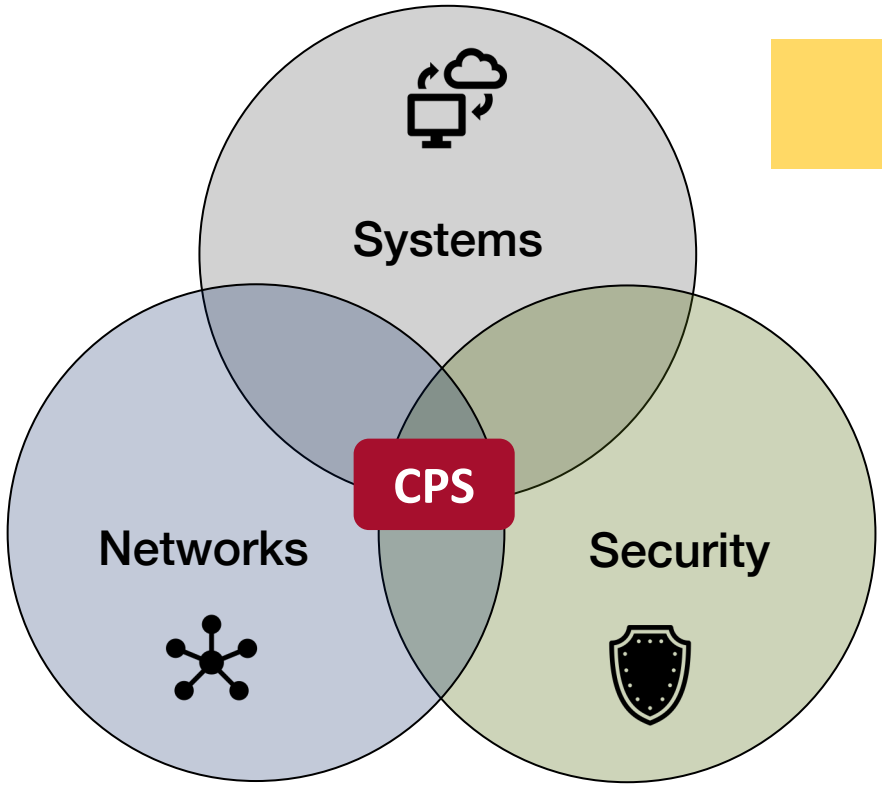
Research Assistant, EECS@WSU

$whoami?

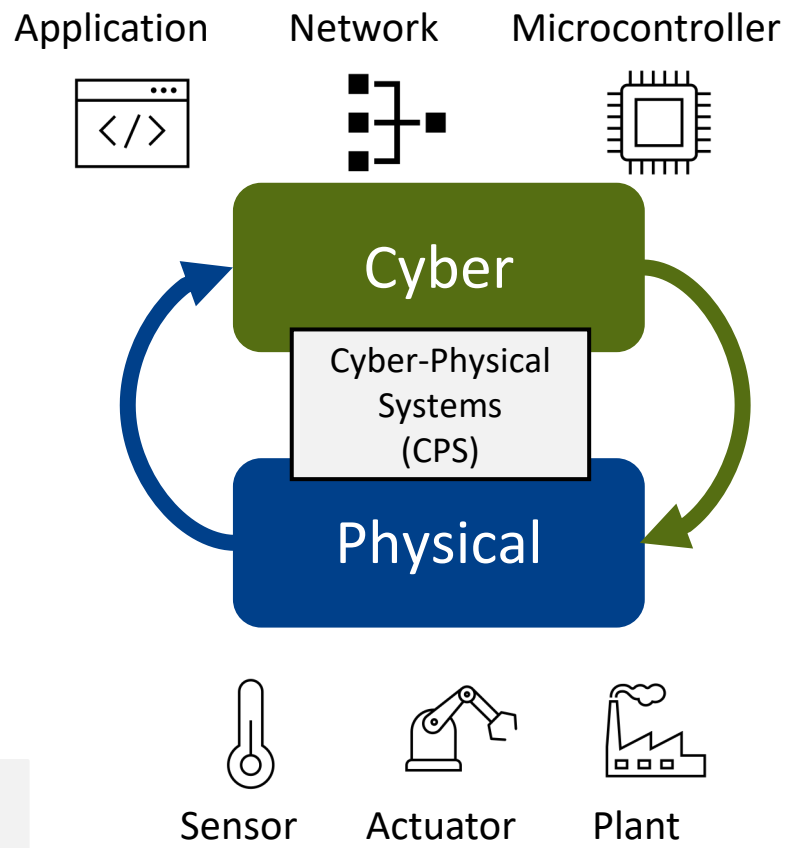
- Assistant Professor
  - EECS@WSU
  - EME B53

- Lab:
  - Cyber-Physical Systems Security Research Lab
  - https://cps2rl.github.io

My Research: Computer Systems



Open research positions
Internet-of-things + Security



CPS Applications: Automobiles, Avionics, Control Systems, Medical Robots, …

Meet the Team

Mohammad Fakhruddin Babar

PhD student

CPS2RL, EECS@WSU

Tamim Ahmed

PhD student

CPS2RL, EECS@WSU

Tasnim Farhan Fatin

PhD student

CPS2RL, EECS@WSU

Agenda

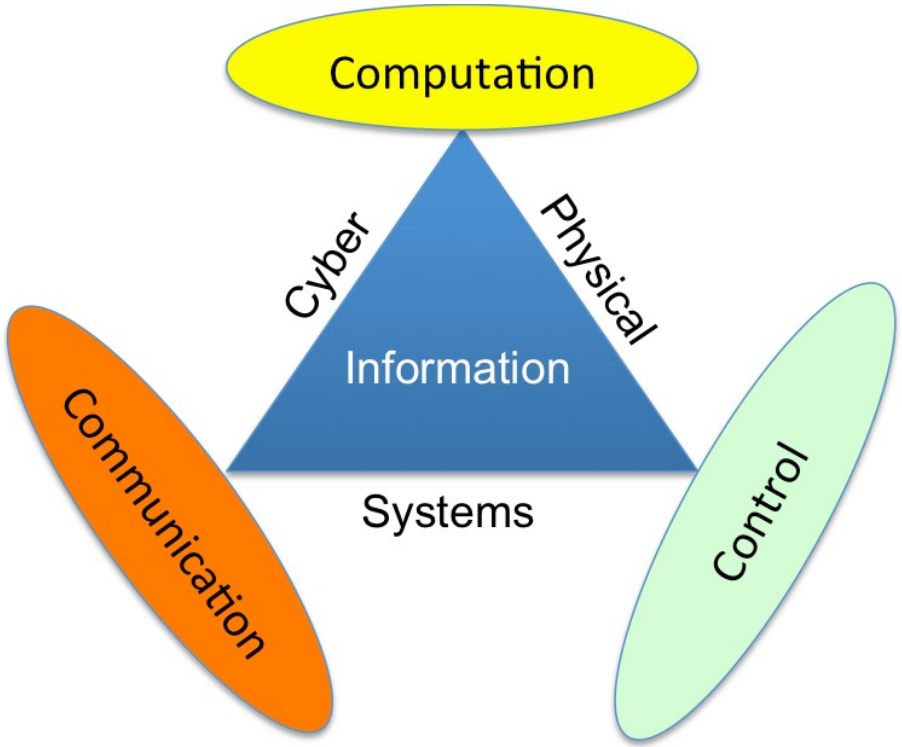
- Cyber-Physical System (CPS) overview
- Hands-on CPS demonstration
- Two Hackathons
  - Competition → teams who solve first win!

What's Next?

- Overview of Cyber-Physical Systems (CPS)
- Security and Resiliency of CPS

Cyber-Physical Systems (CPS)

- Systems with “Cyber” and “Physical” components



Cyber-Physical Systems (CPS)

CYBER

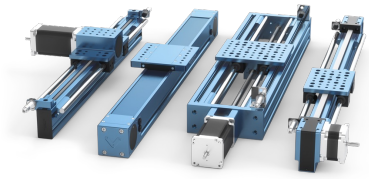
SW, Control Algorithm, Code

Networking

Microcontroller, ECU, PLC

PHYSICAL

Sensors

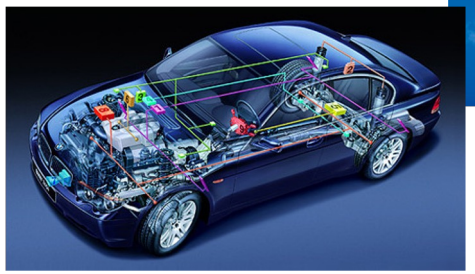


Actuators

Plant

CPS Applications

Automobiles



Avionics

Control Systems

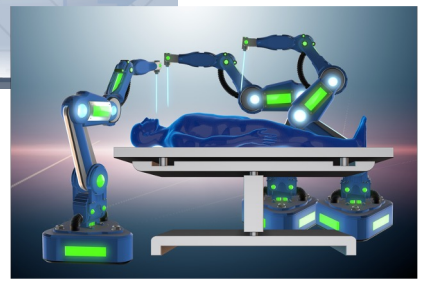
Unmanned Vehicles

Autonomous Driving

Manufacturing

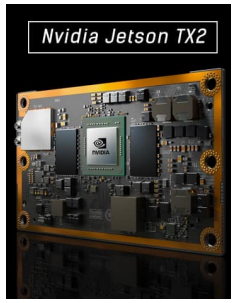
Surveillance

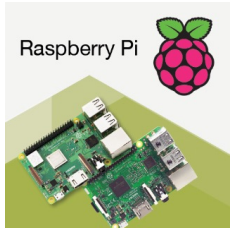
Healthcare



Trends in CPS

- More **powerful** and **cheaper** computing

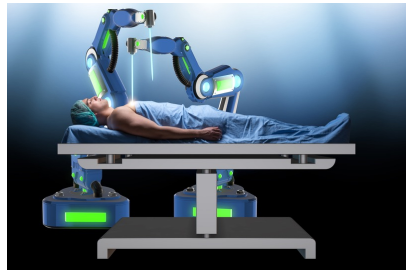




- More **connected**
  - Internet-of-Things

CPS Design Challenges: High Cost of Failures

- Safety-critical: **human life at risk**



- Recalls, production delays, lawsuits, etc.
  - Toyota SUA:
    - 89 deaths and 52 injuries in the USA, $1.2B settlement with DoJ in 2014, lawsuits
  - Ford Mustang Mach-E recall in 2022
    - Software error → unintended acceleration/deceleration, a loss of drive power
  - Boeing 737 MAX was grounded worldwide 2019-2020
  - Accidents involving self-driving cars: Tesla, Uber, …

Safety Failure in Boeing 737 Max

- Lion Air Flight 610 crashed into the Java see in 2018 [1]
- Ethiopian Air 302 crashed after takeoff in 2019 [2]
- Boeing 737 MAX was grounded from March 2019 to November 2020

- Caused by stall prevention system
  - Faulty Maneuvering Characteristics Augmentation System (MCAS)
  - Sensor error → noise down

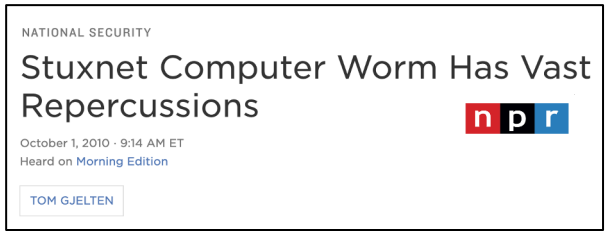
[1] https://en.wikipedia.org/wiki/Lion_Air_Flight_610

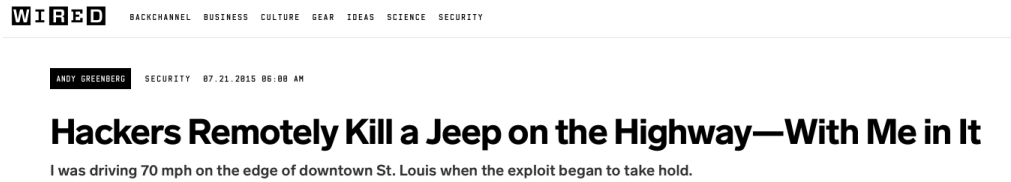
[2] https://en.wikipedia.org/wiki/Ethiopian_Airlines_Flight_302

CPS Security?

CPS Security Threats

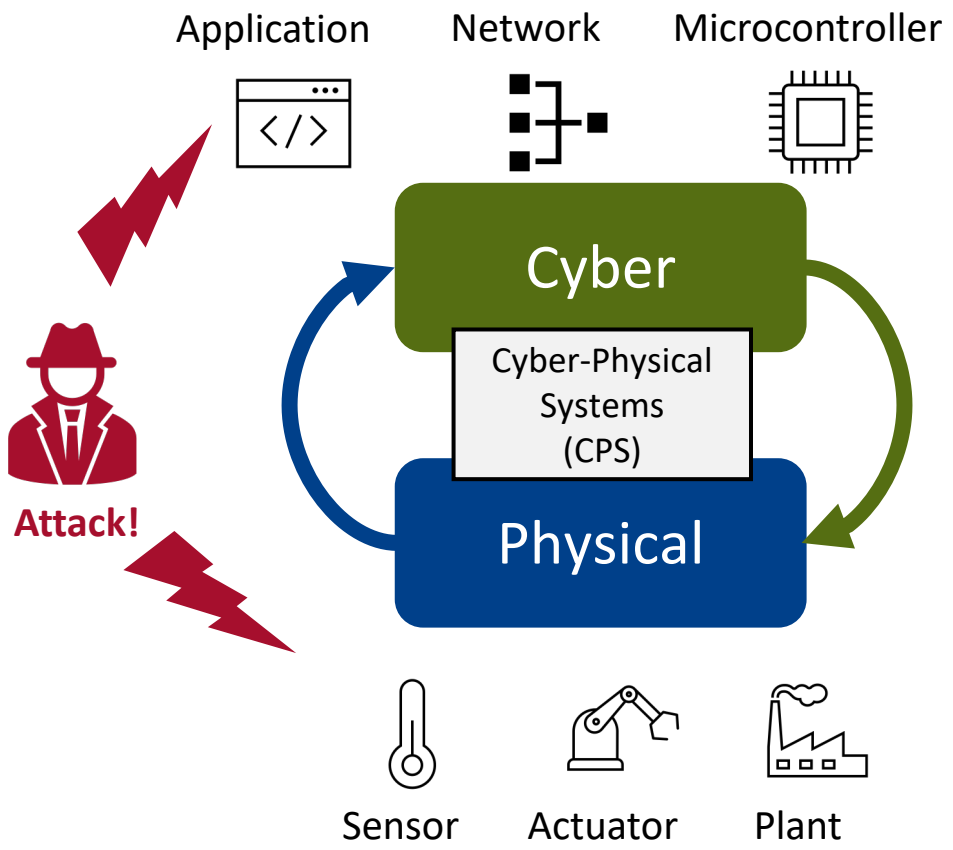
- Modern cyber-physical systems are vulnerable!
- Real-world attacks
  - Stuxnet (control plant attacks)
  - Remote automobile hacking
  - Police drone hacking
  - Medical devices





CPS Security is HARD!

- Why?
  - Cannot simply use conventional, general cyber security schemes to achieve all CPS protections
  - Most CPS security solutions need to be closely integrated with the underlying physical process control features



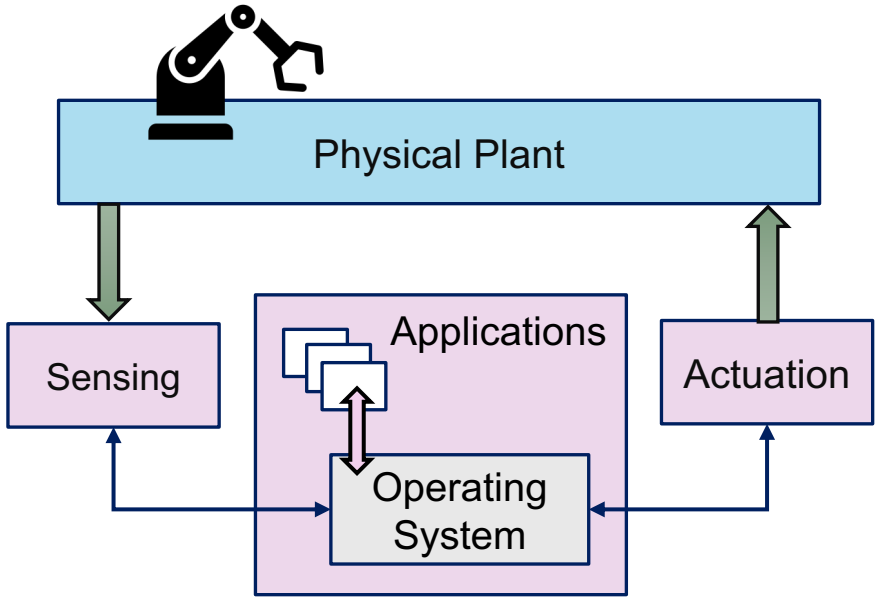
More in CPTS 439!

The Rest of Today's Session

CPS Design and Hackathon

Cyber-Physical System

- Core is sensing and actuation
- What's next?
  - Program actuators
  - Hack actuators



Actuator: Servo

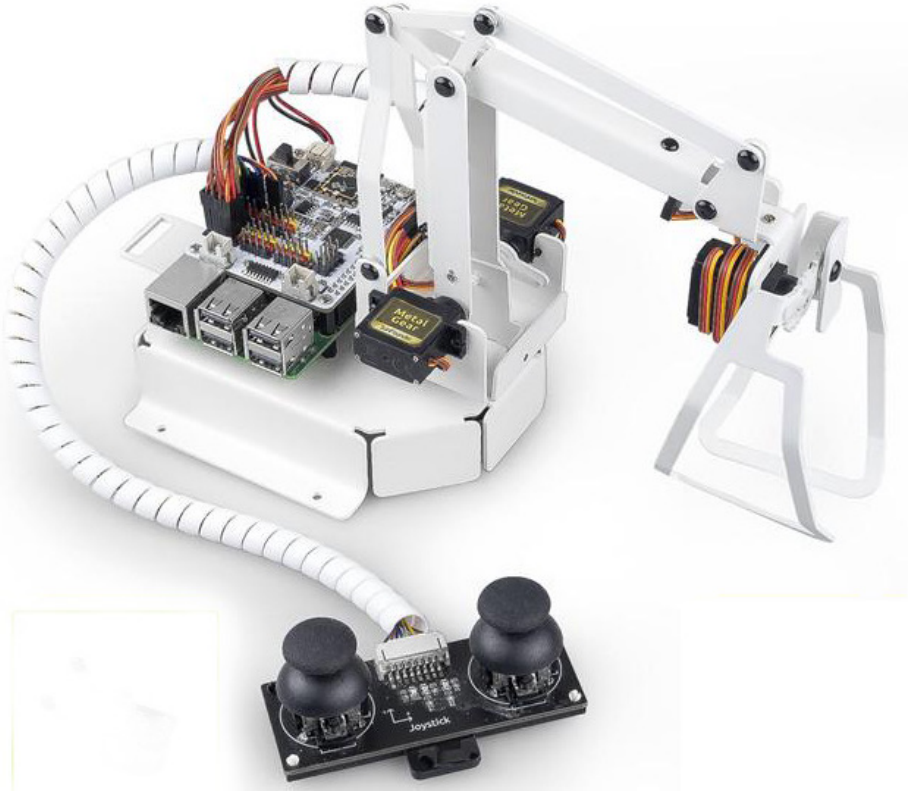
- What is a servo motor?
  - a type of motor that allows for precise control of rotational speed, or velocity
- For our workshop, we are using two types of servo
  - 9G 90 Micro Servo

  - Hiwonder HPS 2027 Servo

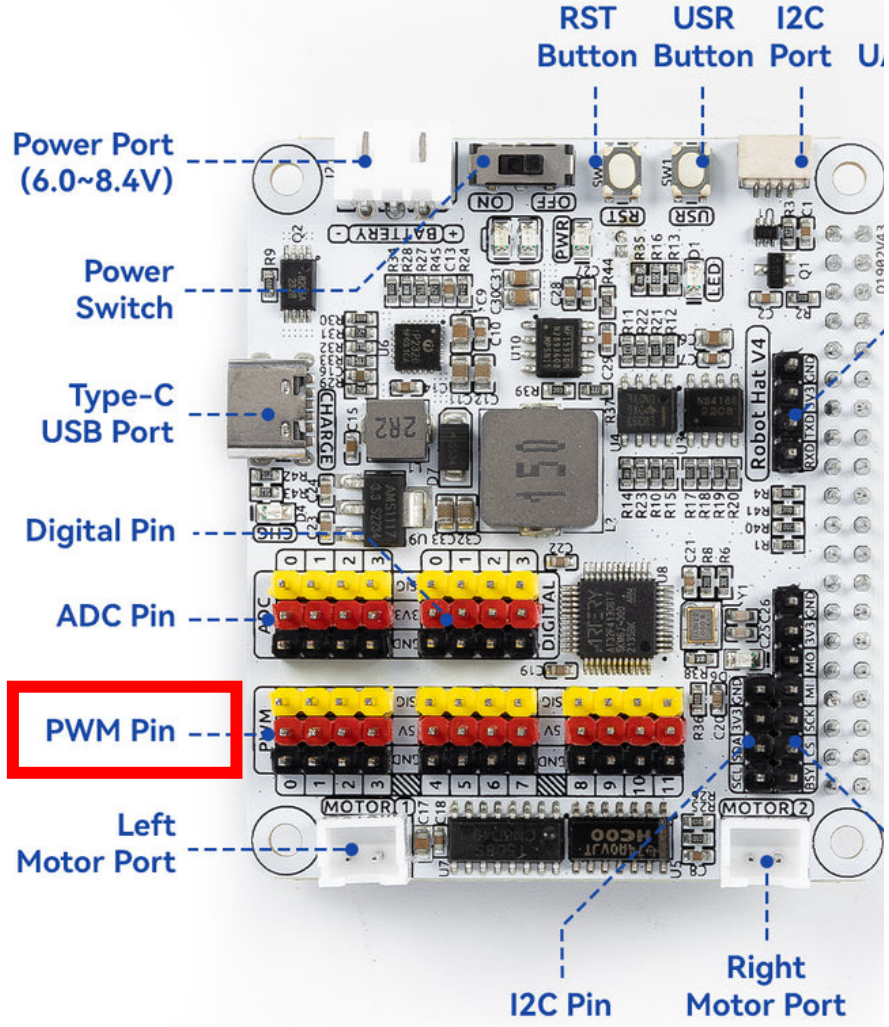
Hardware

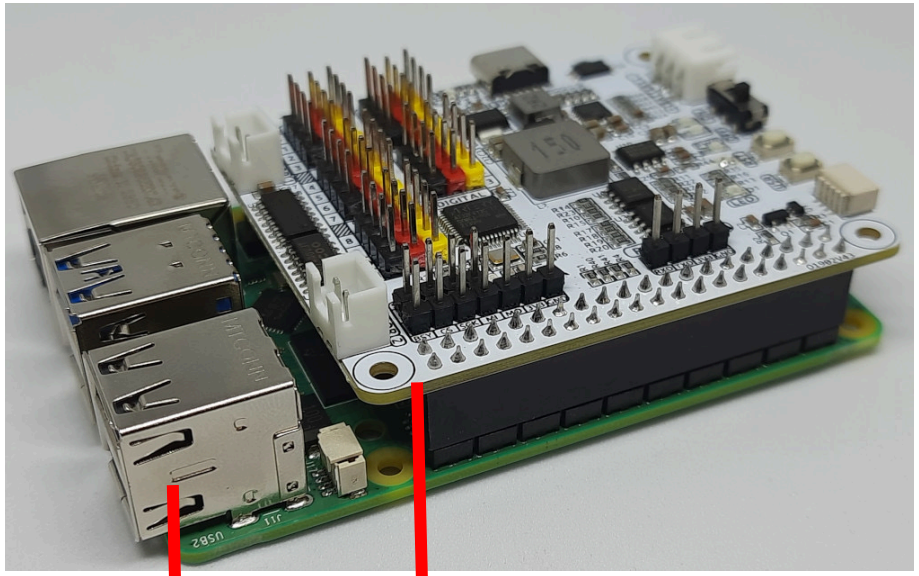
- Screwdriver Set

- PiARM Robot



Servo Control: Hardware Setup





Robot Hat

Raspberry Pi

Servo Control: Example Code

- Importing libraries

```
# Import Servo class from the robot_hat library
from robot_hat import Servo

# Importing the time library to introduce delays
import time
```

- Initialize the servo

```
# Initialize the servo on the appropriate pin
actuation_servo1 = Servo(0) # Servo1 on channel 0
```

Servo Control: Example Code

- Actuation Zero

```
# Function to set actuation_servo to 0 degrees
def actuation_zero():
    actuation_servo1.angle(0) # Move the servo to the 0
    time.sleep(1) # Pause for 1 second after the movement
```

- Actuation Left

```
# Function to move actuation_servo to the left
def actuation_left():
    actuation_servo1.angle(90) # Move the servo to 90 deg
    time.sleep(1) # Pause for 1 second after the movement
```

- Actuation Right

```
# Function to move actuation_servo to the right
def actuation_right():
    actuation_servo1.angle(-90) # Move servo to -90 deg
    time.sleep(1) # Pause for 1 second after the movement
```

Demonstration

Hackathon Problem #1 (45 minutes)

Hackathon Problem #1 (45 minutes)

Task A — Servo Angle Control [10 Minutes]:

- You are provided with a servo motor and a Raspberry Pi
- You are required to write a program that precisely controls the servo to move to a set of given angles (30°, 90°, and -45°) in sequence

Task B — Servo Motion Control [15 Minutes]:

- In real-world applications, rapid servo movements can lead to mechanical wear or damage
- You are required to modify the previous solution (Task A) to ensure that the servo moves gradually and smoothly between angles, simulating a slow-motion effect

Task C — Implement a Denial-of-Service (DoS) Attack [20 Minutes]:

- In real-world environment, malicious users may abuse input systems to disrupt normal operation
- You are required to implement a "Denial of Service" attack
  - In this scenario, a user gives input angle position and servo will move to that position
  - However, by pressing the key 'L', the system should freeze and ignore further inputs

(Possible) Solution

Hackathon Problem #1 (Solution)

- **Task A — Servo Angle Control :** You are provided with a servo motor and a Raspberry Pi. You are required to write a program that precisely controls the servo to move to a set of given angles (30°, 90°, and -45°) in sequence.

```
# Import the Servo class from the robot_hat library
from robot_hat import Servo
# Importing the time library to introduce delays
import time

# Initialize the servo on the appropriate pin
actuation_servo1 = Servo(0) # Servo1 on channel 0 for slower movement

actuation_servo1.angle(30)
time.sleep(1)
actuation_servo1.angle(90)
time.sleep(1)
actuation_servo1.angle(-45)
time.sleep(1)
```

Hackathon Problem #1 (Solution)

Task B — Servo Motion Control: In real-world applications, rapid servo movements can lead to mechanical wear or damage. You are required to modify the previous solution (Task A) to ensure that the servo moves gradually and smoothly between angles, simulating a slow-motion effect.

```
# Import the Servo class from the robot_hat library
from robot_hat import Servo
# Importing the time library to introduce delays
import time

# Initialize the servo on the appropriate pin
actuation_servo2 = Servo(0) # Servo2 on channel 0 for faster movements

# Function to move actuation_servo smoothly
def slow_move_left():
    for i in range(-90, 90):  # Looping from -90 to 90 degrees
        actuation_servo2.angle(i)
        time.sleep(0.01)
    time.sleep(1)

# Run function to start the movement
slow_move_left()  # Move the servo Left
```

Hackathon Problem #1 (Solution)

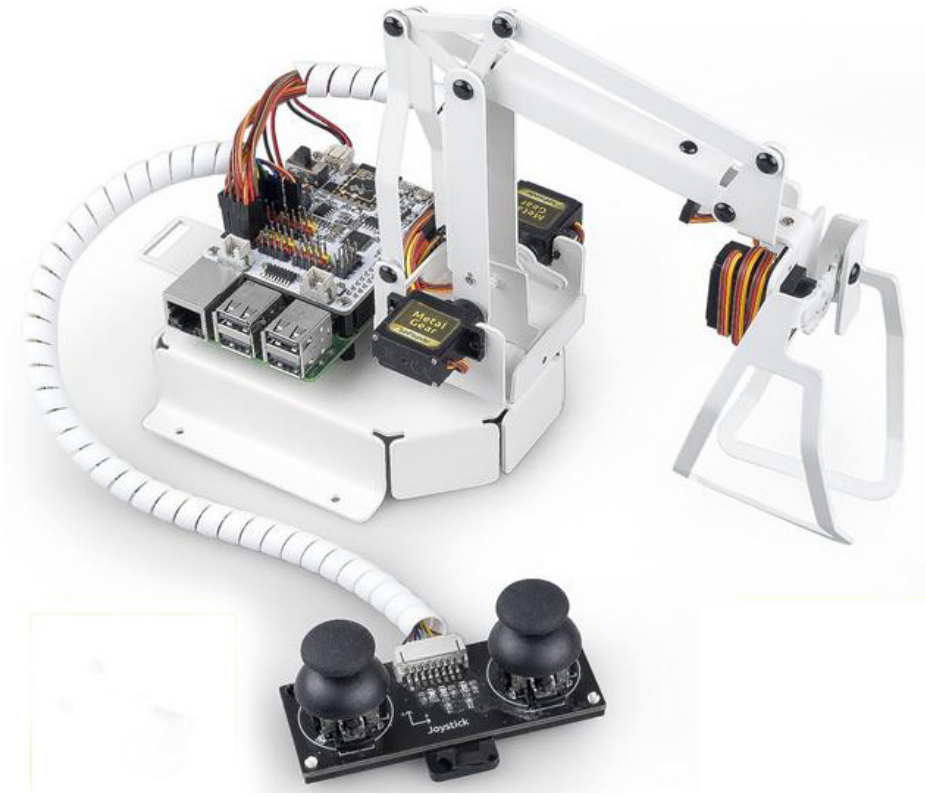
Task C — Implement a Denial-of-Service (DoS) Attack : In real-world environment, malicious users may abuse input systems to disrupt normal operation. You are required to implement a "Denial of Service" attack. In this scenario, a user gives input angle position and servo will move to that position. However, by pressing the key 'L', the system should freeze and ignore further inputs. Here, system freeze refers freezing the servo motors.

```
from robot_hat import Servo
import time
# Initialize the servo on channel 0
actuation_servo2 = Servo(0)
# Function to move actuation_servo2 based on user input
def attack():
    while True:
        user_input = input("Enter angle (0-180) or 'L' to loop input mode: ")
        if user_input == "L":
            print("Servo is Freezed")
            while True:
                angle_input = input("Enter angle = ")
                time.sleep(0.5)
        actuation_servo2.angle(int(user_input))
# Start the attack function
attack()
```

PiARM Robot

PiARM Robot Kit

- 4 Servos
- Angle Mode: Writes a certain angle to the three servos on the arm, thus rotating the arm to a specific position



- Hardware:
  - PiARM Robot
  - Raspberry Pi
  - Robot HAT
  - Battery

PiARM Robot: Simple Task Example

- Command to Robot
  - Voice command
  - Simple Command: Left, Right, Center, Up, Down, Open, Close
  - VOSK Speech recognition model
    - Takes Audio Input
    - Convert the input into string

VOSK Download Link: https://alphacephei.com/vosk/models

PiARM Robot: Code

- Import Libraries

```
from robot_hat import Servo
import time
from voice import recognize_speech
```

- Initialize the servo

```
# Servo Initialization
wrist_servo = Servo(4) # Controls wrist (up/down movement)
arm_servo = Servo(5)   # Controls arm (forward/backward movement)
base_servo = Servo(6)  # Controls base rotation (left/right/center)
grip_servo = Servo(7)  # Controls grip (open/close for holding)
```

PiARM Robot: Code

- Grip Servo → Open

```
# Function to open the grip by gradually decreasing the servo angle
def grip_open(servo):
    print("Grip Open")  # Message indicating the grip is opening
    for i in range(55, 0, -1):
        servo.angle(i)  # Set servo to decreasing angle
        time.sleep(0.009)  # Short delay for smooth transition
    time.sleep(0.5)  # Pause for 0.5 second after opening
```

- Grip Servo → Close

```
# Function to close the grip by gradually increasing the servo
angle
def grip_close(servo):
    print("Grip Close")  # Message indicating the grip is closing
    for i in range(0, 55):
        servo.angle(i)  # Set servo to increasing angle
        time.sleep(0.009)  # Short delay for smooth transition
    time.sleep(0.5)  # Pause for 0.5 second after closing
```

PiARM Robot: Code

- Base Servo → Center Position

```
# Function to move base
def base_move_center(servo):
    print("Base location: Center")  #indicating base is centered
    servo.angle(0)  # Set servo to center position
    time.sleep(1)  # Pause for 1 second
```

- Wrist Servo → Up

```
# Function to move the servo claw upwards
def move_wrist_up(servo):
    print("Moving Up")  # indicating upward movement
    for i in range(10, 80):
        servo.angle(i)  # incrementally moving it up)
        time.sleep(0.009)  # Small delay to allow smooth motion
    time.sleep(0.5)  # Pause for 0.5 second after reaching the top
```

Demo: Pick an Object

- Command to Robot
  - Voice command
  - Simple Command: Left, Right, Center, Up, Down, Open, Close
  - VOSK Speech recognition model

Demonstration

Hackathon Problem #2 (60 minutes)

Hackathon Problem #2 (1 hour)

Task A — Robot Control [30 Minutes]:

- You are given a PiARM Robot
- You are required to control the robot arm using voice commands (up, down, left, right, open, close) to pick up an object from the left side and place it in a designated position on the right side

Task B — Attack the Robot [30 Minutes]:

- Your task is to demonstrate a “logic bomb attack” within the Task A scenario after the robot picks up an object from the left side
- The attacker (you) must issue a crafted voice command that triggers the logic bomb
  - You can be creative to inject your own voice command (e.g., fakeleft)
  - This should cause the robot to destabilize and fail to complete the task (move object)