SYSTEM MODELING

Search and rescue robot

Technobot

Theophile THOMAS

Alexandre MENSAH

Alexandre EANG

Zhipeng ZENG

Xiaosen CHEN

04/12/2023

Primary Goal:

Develop a robot capable of autonomously searching, finding, and rescuing a specific target.

Design Specifications:

Divided into three levels - Bronze, Silver, and Gold, with increasing complexity.

- The Bronze level includes basic autonomous exploration and target detection;
- The Silver level adds target pick-up and placement capabilities;
- The Gold level incorporates a remote system interface (GUI).

System Architecture:

The robot is based on the EV3 core system, encompassing modules for direction control, grabbing, pick-and-place, and telemetry.

Team introduction:

- Theophile THOMAS - Project Manager
- Alexandre MENSAH - Test Chief Engineer
- Alexandre EANG - Documentation manager
- Zhipeng ZENG - Software Chief Engineer
- Xiaosen CHEN - Hardware Chief Engineer

Team Advantages:

Expertise Across the Board

Collaborative Synergy

Commitment to Quality

Design input definition

Main problem

Create a robot that can search, find and rescue a defined object.

Test conditions (same for all levels)

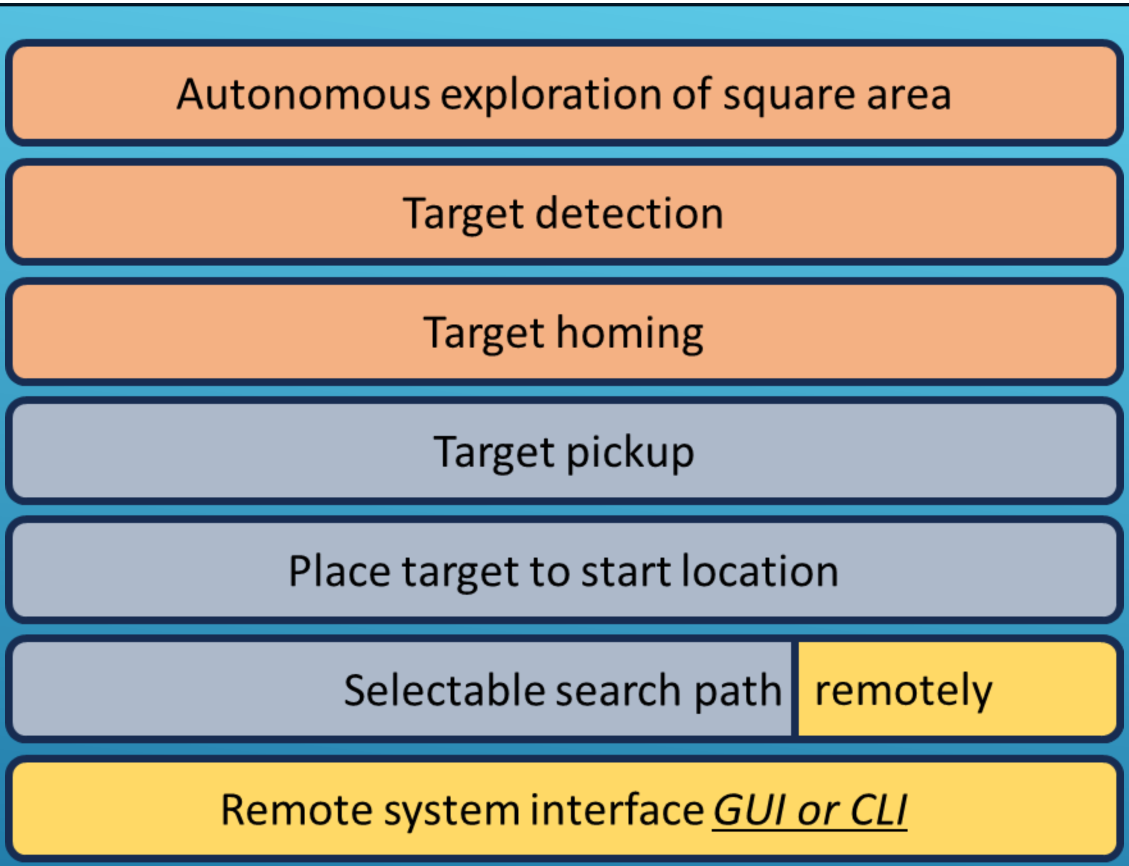
- Ground area 1.5m x 1.5m square
- Area delimited by ***black tape*** *on ground*
- 1 target dropped inside area : ***cylindrical object***
- Robot starts at one corner of the area

DESIGN INPUT DEFINITION

From ideas & instructions to system specifications

Design input definition

Specifications in function of level



Autonomous exploration of square area

Target detection

Target homing

Target pickup

Place target to start location

Selectable search path remotely

Remote system interface *GUI or CLI*

Design input definition

Validation criteria in function of level

Robot shall explore area with 1 predefined search pattern .. | ..among 3 existing patterns | ..selected remotely

Robot shall make use of a *cartesian* coordinate system

Robot shall set origin to start point

Robot shall detect the target and register its position

Robot shall communicate target position to user through *sound, remote interface or built -in display*

Robot shall return home after detecting target

Robot shall start moving the target home once detected

X

Robot shall communicate its own coordinates to a remote *GUI or CLI*

X

Robot shall communicate its *machine state* to a remote *GUI or CLI*

X

Robot shall asynchronously interpret commands from the remote *GUI or CLI*

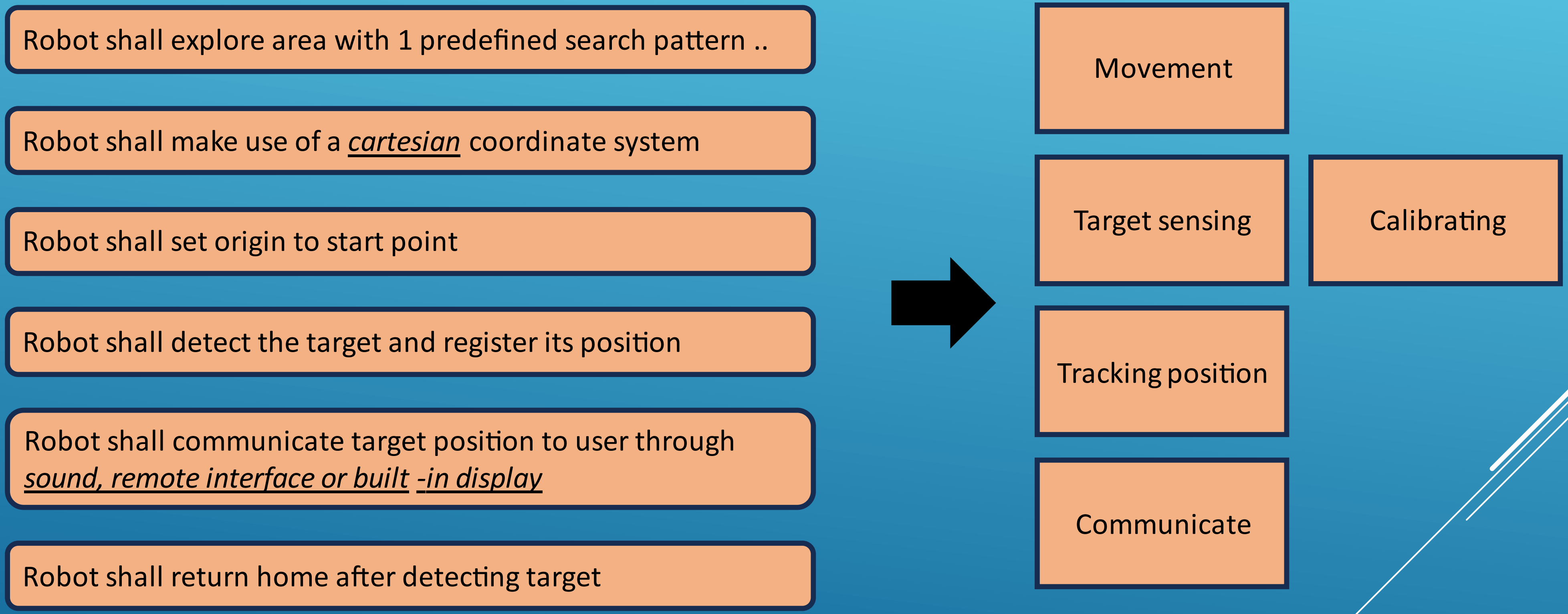
SYSTEM ARCHITECTURE

Structuring the end product block by block

General system architecture : functional view

Bronze validation criteria :

Functions of the bronze system:



Robot shall explore area with 1 predefined search pattern ..

Robot shall make use of a *cartesian* coordinate system

Robot shall set origin to start point

Robot shall detect the target and register its position

Robot shall communicate target position to user through *sound, remote interface or built -in display*

Robot shall return home after detecting target

Movement

Target sensing

Calibrating

Tracking position

Communicate

General system architecture : functional view

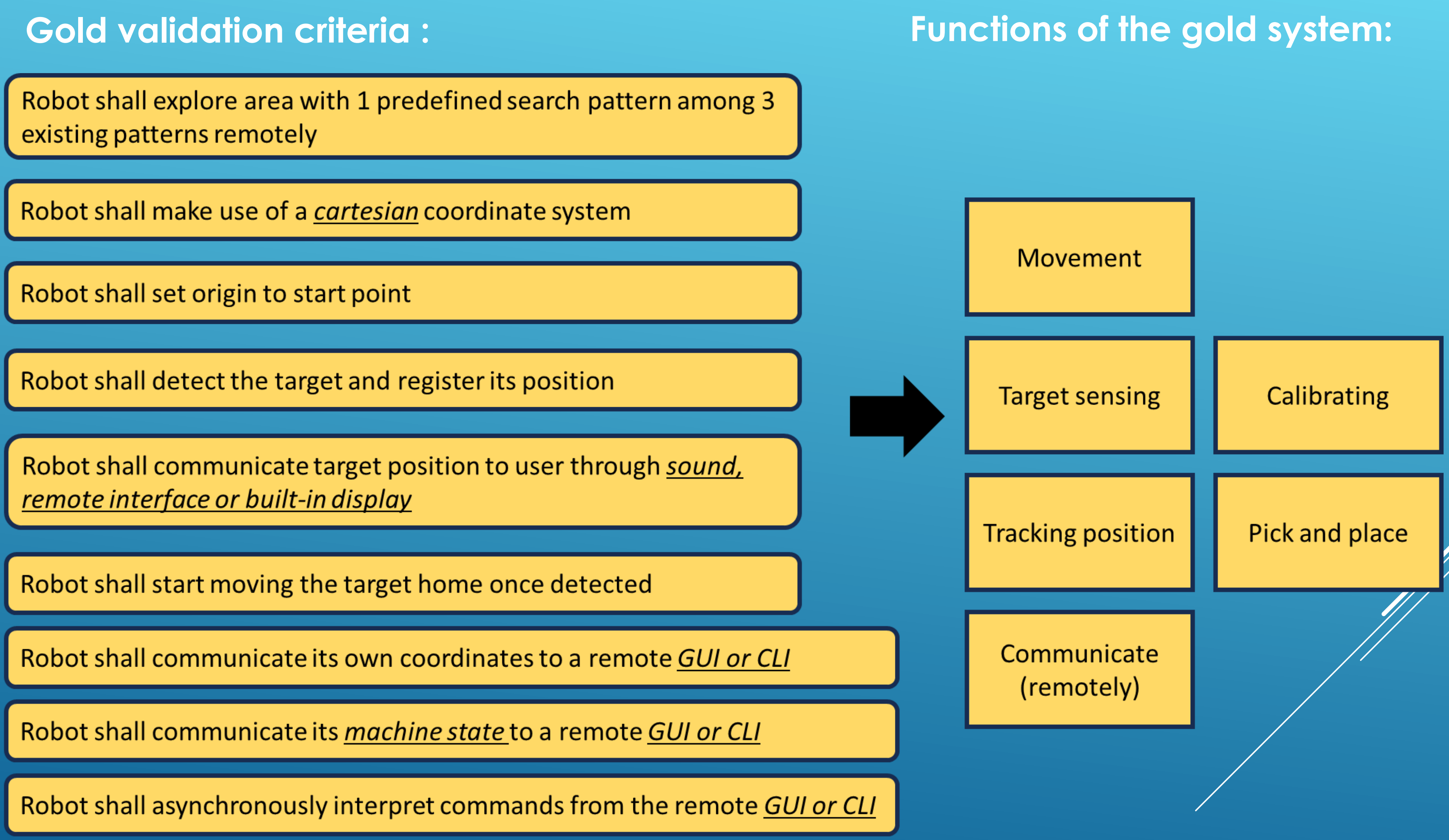
Silver validation criteria :

- Robot shall explore area with 1 predefined search pattern among 3 existing patterns
- Robot shall make use of a *cartesian* coordinate system
- Robot shall set origin to start point
- Robot shall detect the target and register its position
- Robot shall communicate target position to user through *sound, remote interface or built-in display*
- Robot shall start moving the target home once detected

Functions of the silver system:

- Movement
- Target sensing
- Calibrating
- Tracking position
- Pick and place
- Communicate

General system architecture : functional view



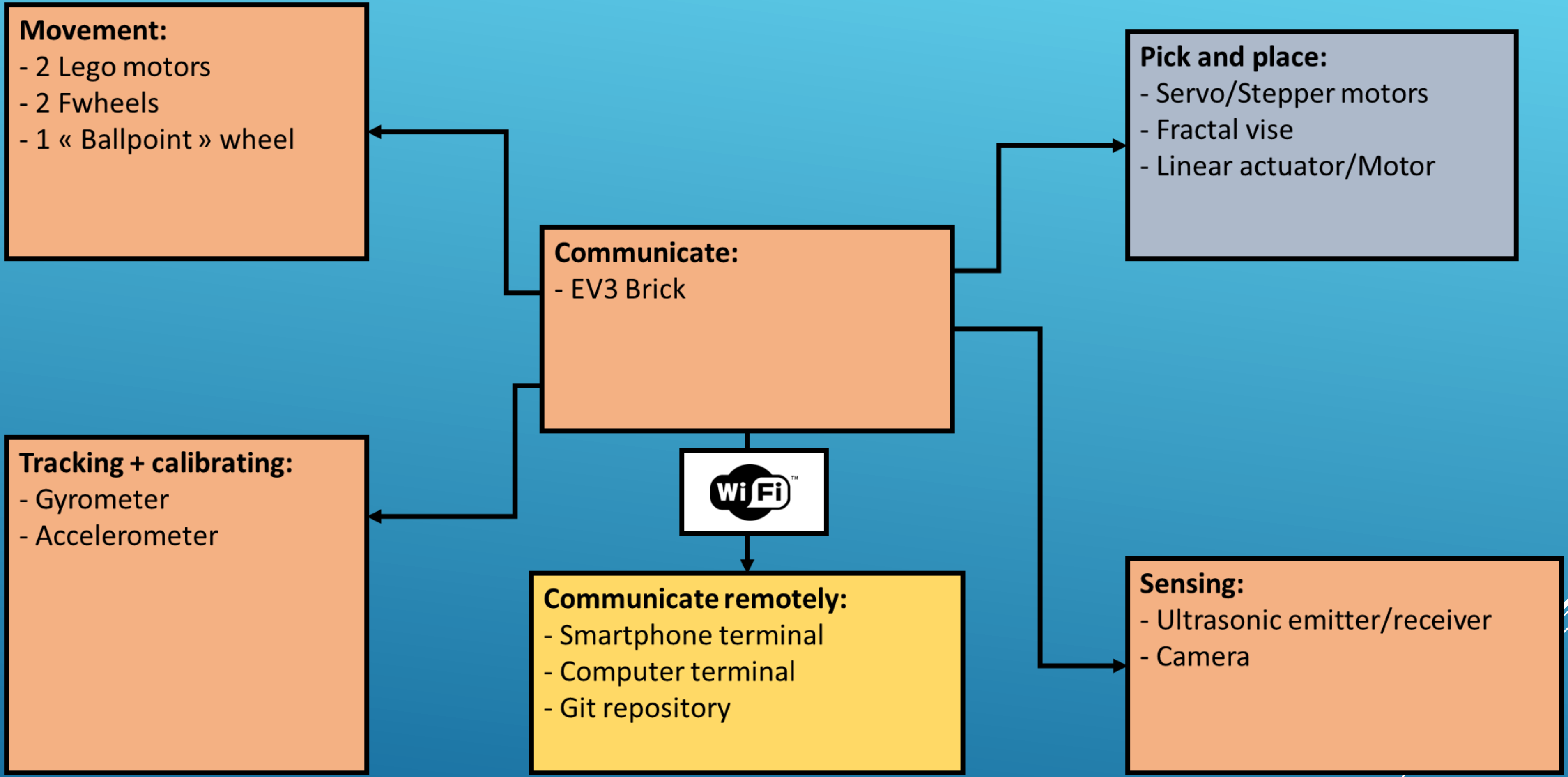
Gold validation criteria :

- Robot shall explore area with 1 predefined search pattern among 3 existing patterns remotely
- Robot shall make use of a *cartesian* coordinate system
- Robot shall set origin to start point
- Robot shall detect the target and register its position
- Robot shall communicate target position to user through *sound, remote interface or built-in display*
- Robot shall start moving the target home once detected
- Robot shall communicate its own coordinates to a remote *GUI or CLI*
- Robot shall communicate its *machine state* to a remote *GUI or CLI*
- Robot shall asynchronously interpret commands from the remote *GUI or CLI*

Functions of the gold system:

- Movement
- Target sensing
- Calibrating
- Tracking position
- Pick and place
- Communicate (remotely)

General system architecture : functional view



Movement:
- 2 Lego motors
- 2 Fwheels
- 1 « Ballpoint » wheel

Communicate:
- EV3 Brick

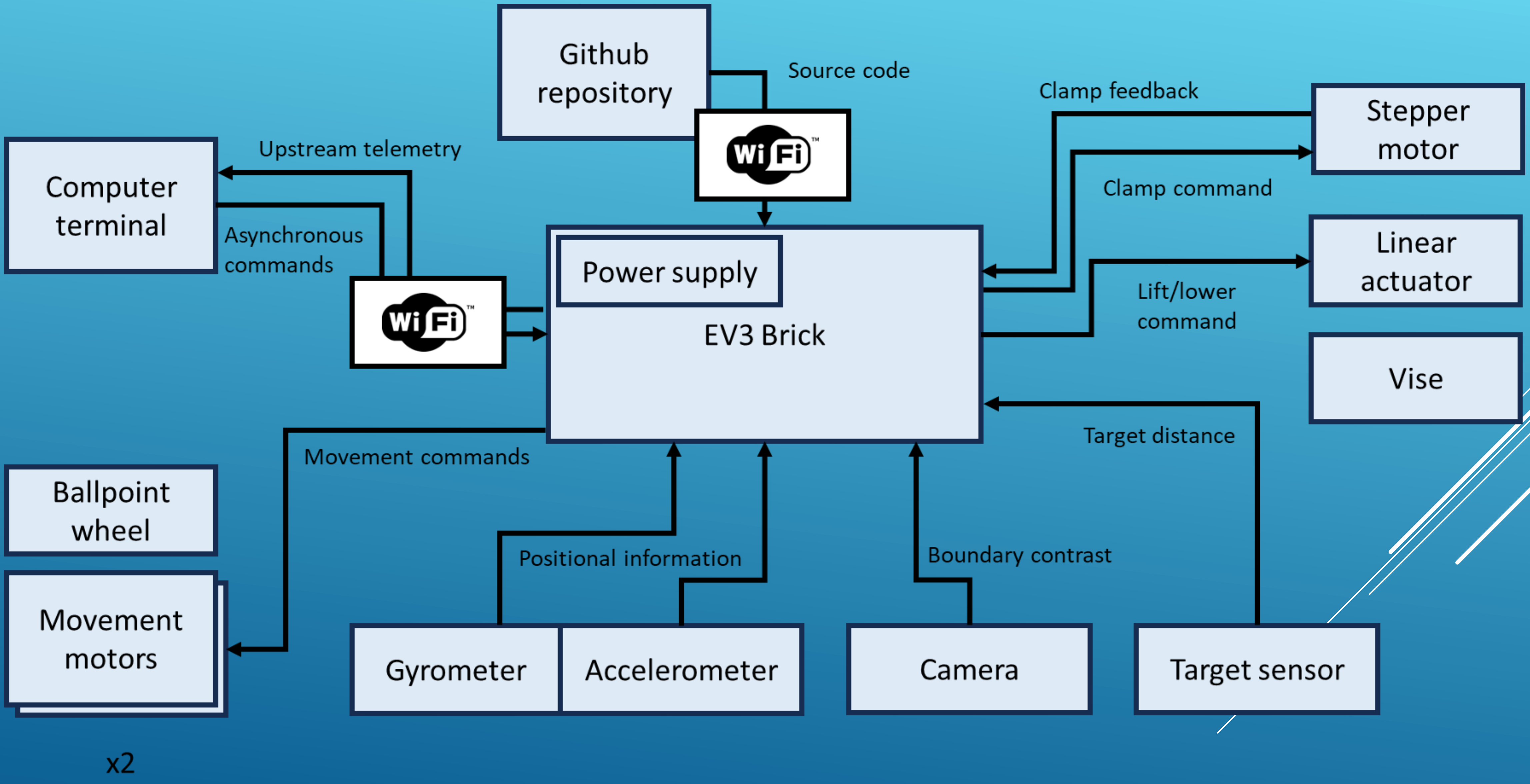
Pick and place:
- Servo/Stepper motors
- Fractal vise
- Linear actuator/Motor

Tracking + calibrating:
- Gyrometer
- Accelerometer

Communicate remotely:
- Smartphone terminal
- Computer terminal
- Git repository

Sensing:
- Ultrasonic emitter/receiver
- Camera

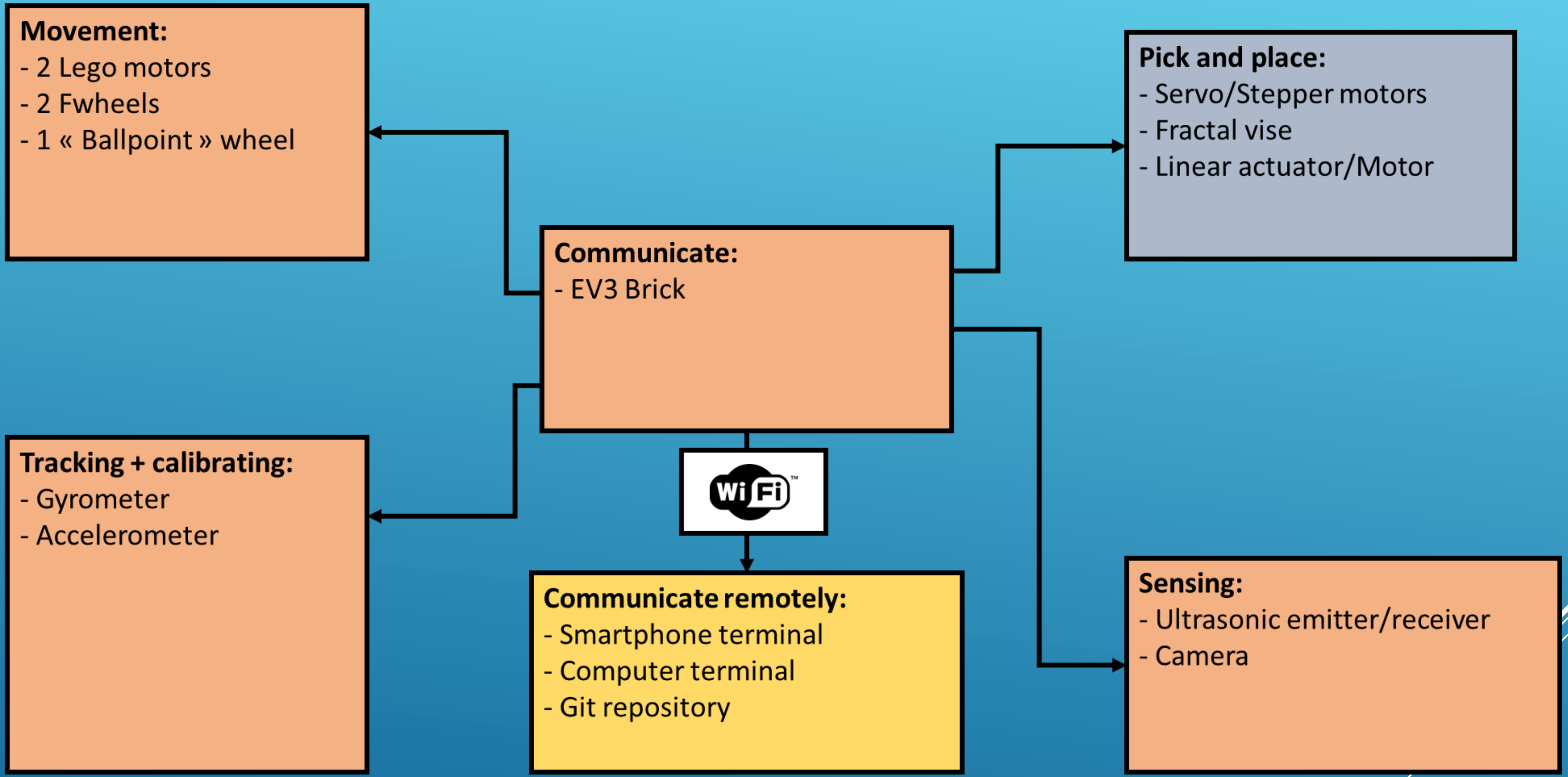
General system architecture : product view



PERIPHERAL BENCHMARK

Providing appropriate solutions to the requirements

Peripheral benchmark : General architecture



Movement:
- 2 Lego motors
- 2 Fwheels
- 1 « Ballpoint » wheel

Pick and place:
- Servo/Stepper motors
- Fractal vise
- Linear actuator/Motor

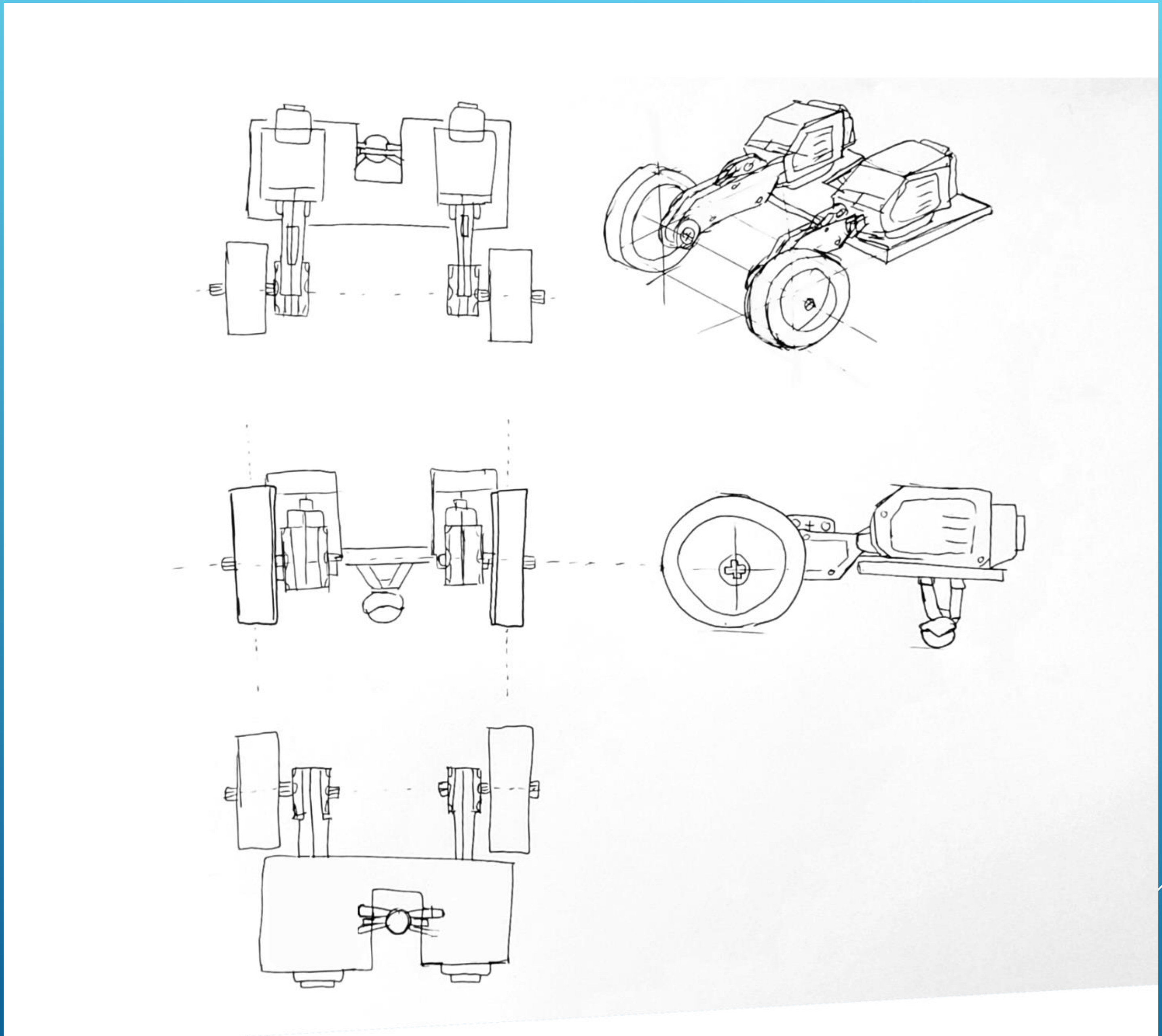
Communicate:
- EV3 Brick

Tracking + calibrating:
- Gyrometer
- Accelerometer

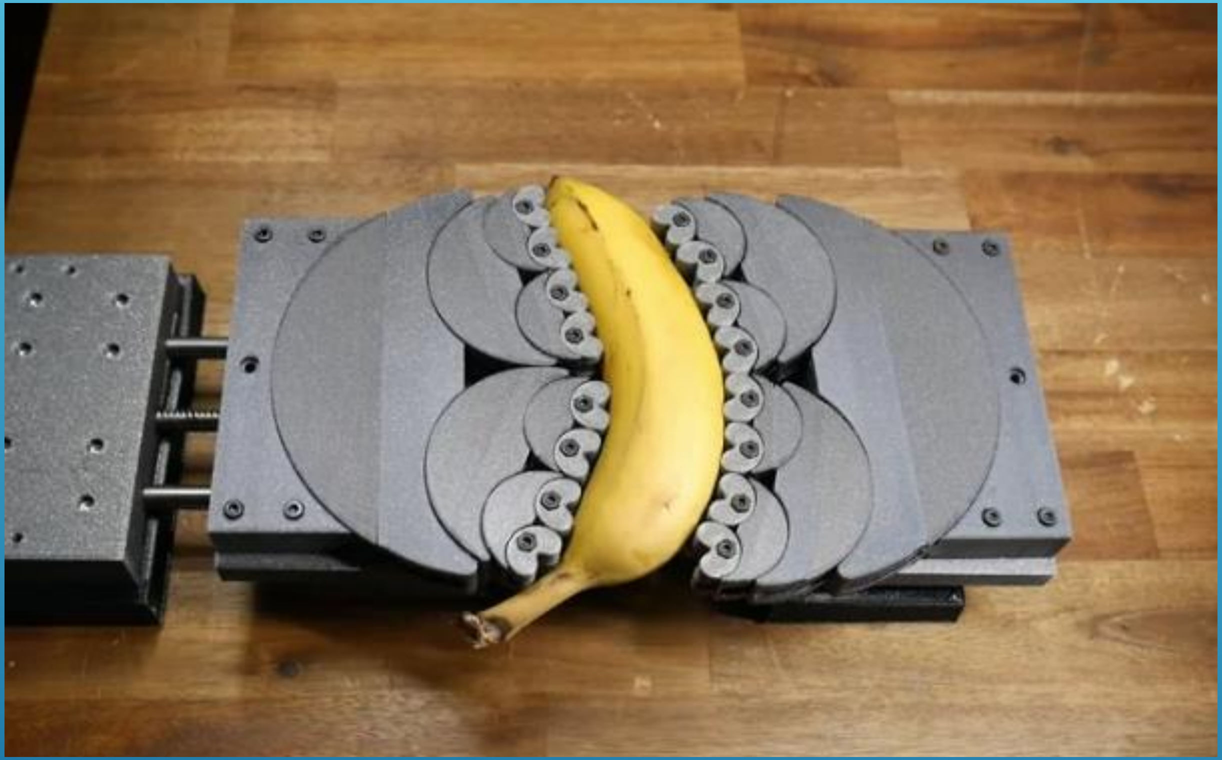
Communicate remotely:
- Smartphone terminal
- Computer terminal
- Git repository

Sensing:
- Ultrasonic emitter/receiver
- Camera

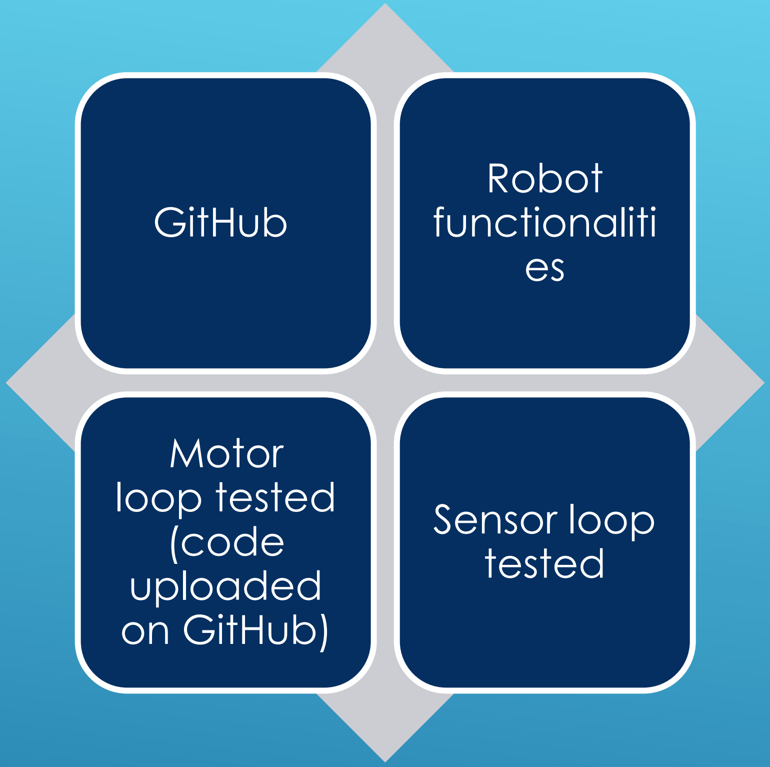
Peripheral benchmark : Movement + Tracking/Calibrating + Sensing



Peripheral benchmark : Fractal vise



ACHIEVEMENTS



GitHub

Robot functionalities

Motor loop tested (code uploaded on GitHub)

Sensor loop tested

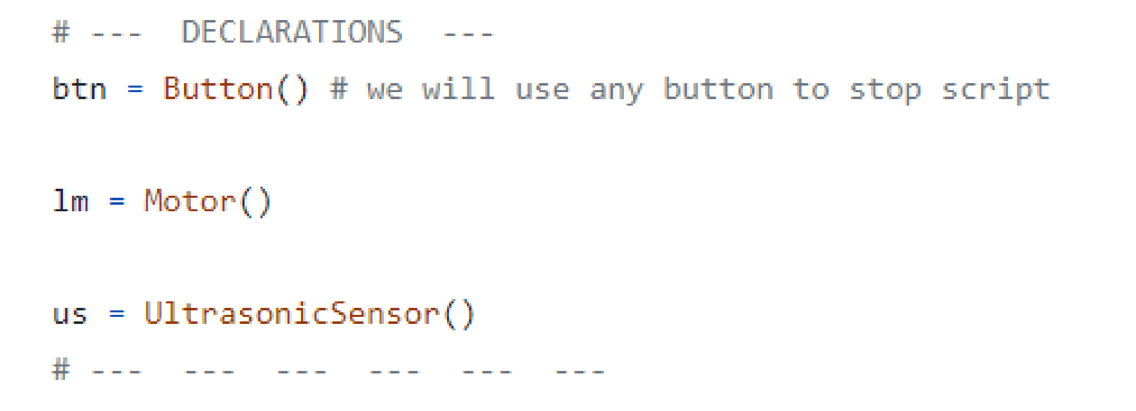
TESTS

Motor-sensor loop code:

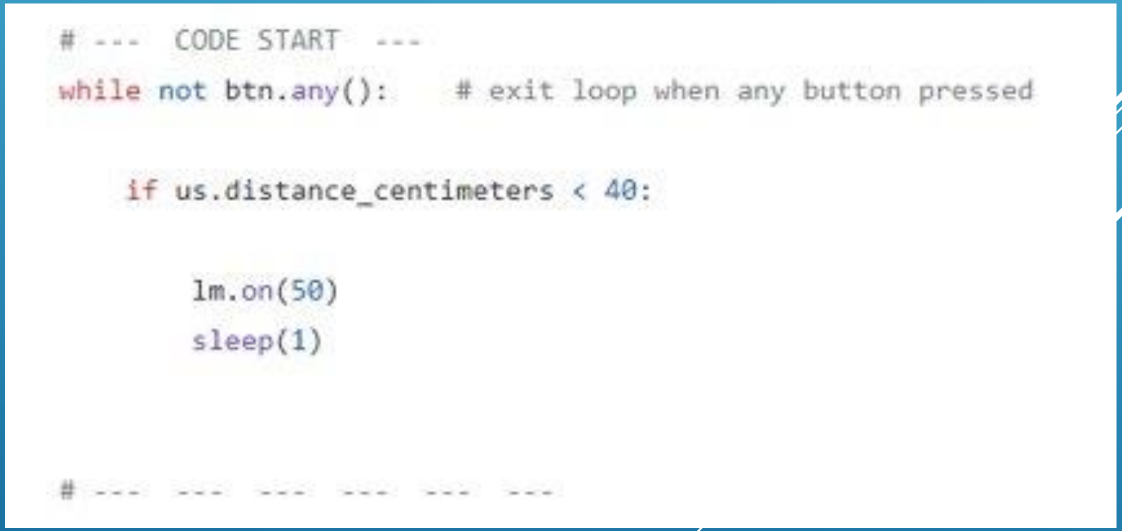
- Start the motor after a sensor information

- Know the basics of moving and detecting

Large Motor
+ Lets you program precise and powerful robotic action.

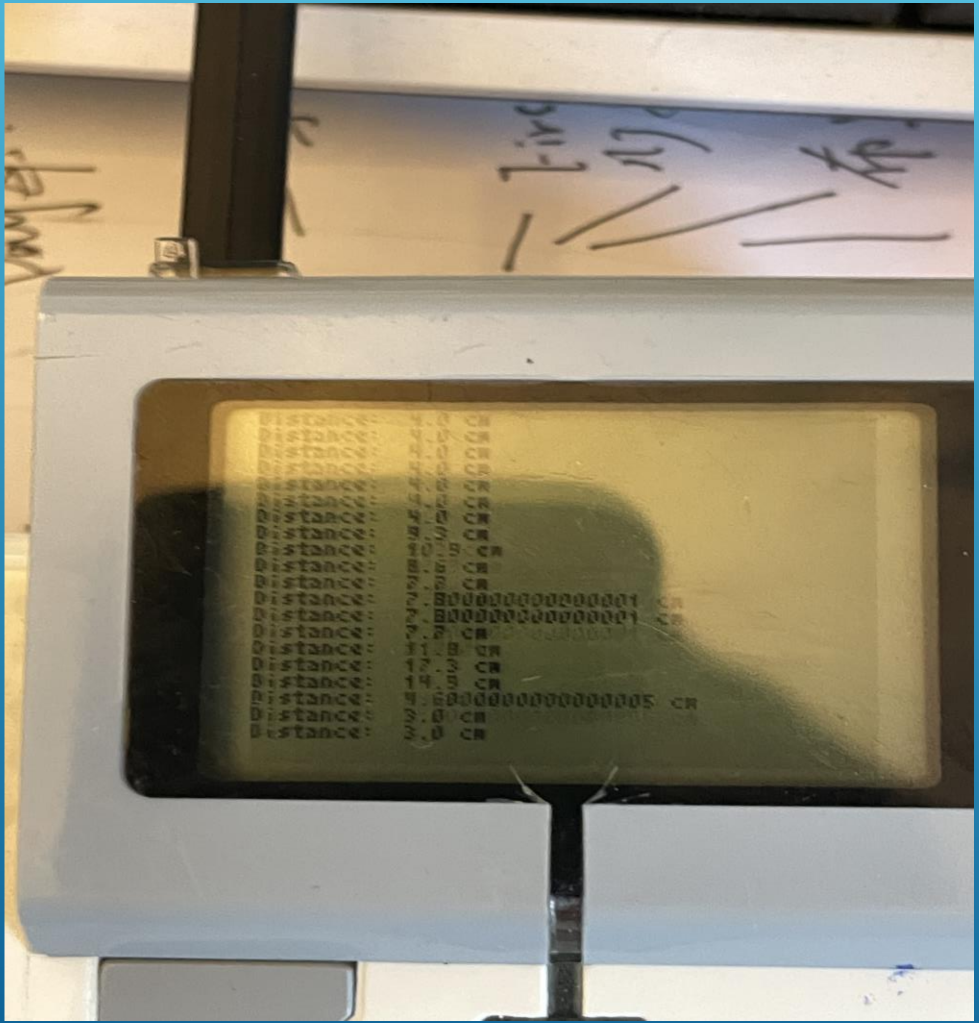


```
# ---  DECLARATIONS  ---
btn = Button() # we will use any button to stop script

lm = Motor()

us = UltrasonicSensor()
# ---  ---  ---  ---  ---  ---
```



```
# ---  CODE START  ---
while not btn.any():    # exit loop when any button pressed

    if us.distance_centimeters < 40:

        lm.on(50)
        sleep(1)


# ---  ---  ---  ---  ---  ---
```

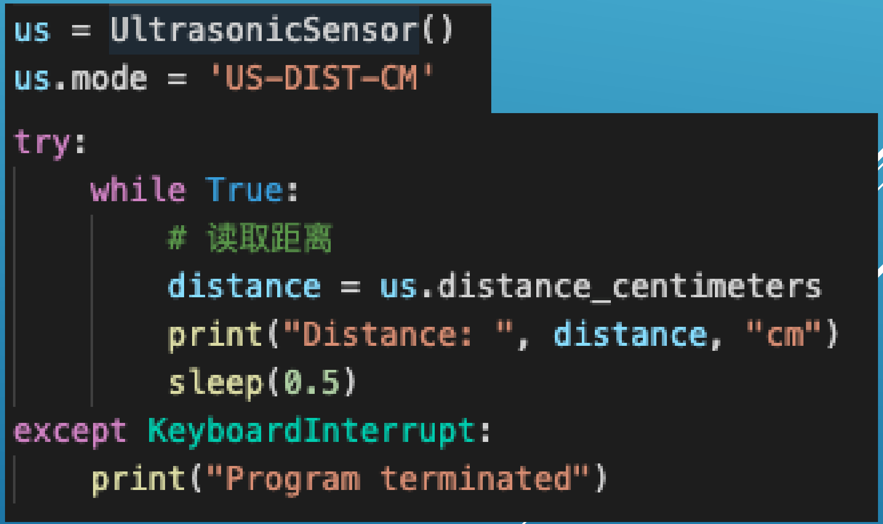
Ultrasonic Sensor



```
Distance:  4.0 cm
Distance:  4.0 cm
Distance:  4.0 cm
Distance:  4.0 cm
Distance:  4.0 cm
Distance:  4.0 cm
Distance:  4.0 cm
Distance:  9.3 cm
Distance:  10.9 cm
Distance:  8.6 cm
Distance:  7.8 cm
Distance:  7.800000000000001 cm
Distance:  7.800000000000001 cm
Distance:  7.8 cm
Distance:  11.8 cm
Distance:  17.3 cm
Distance:  14.9 cm
Distance:  9.600000000000005 cm
Distance:  3.0 cm
Distance:  3.0 cm
```

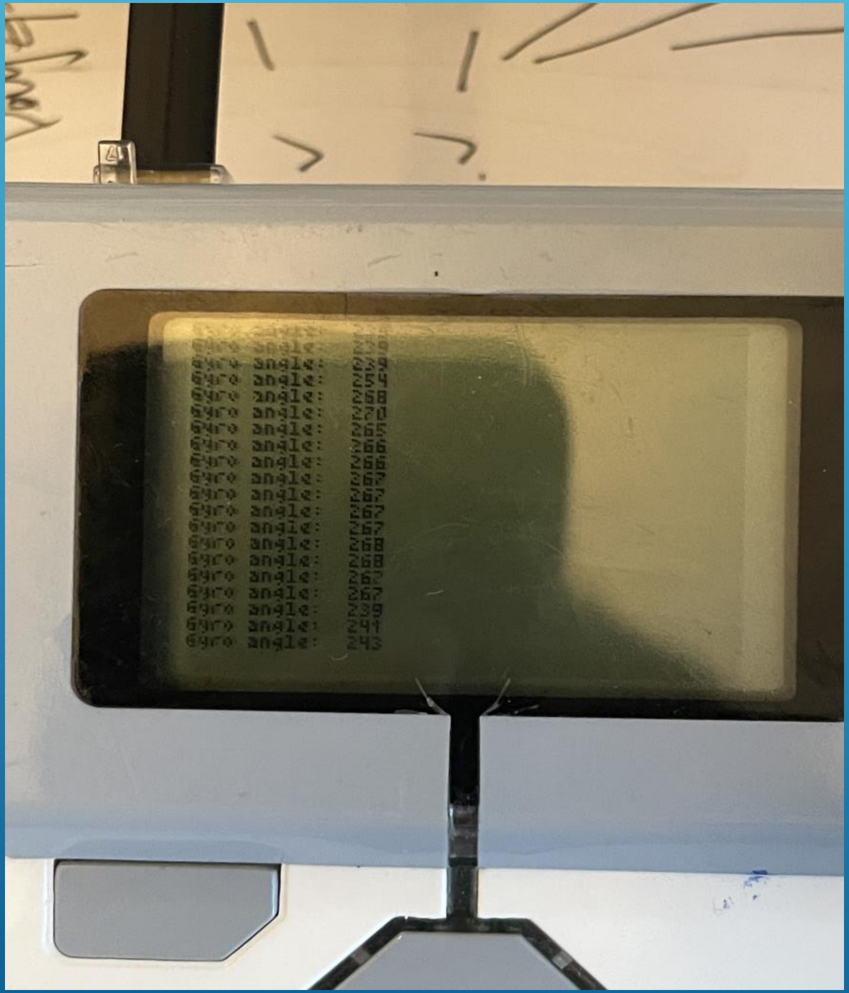
Ultrasonic Sensor

- Uses reflected sound waves to measure distance between the sensor and any objects in its path.

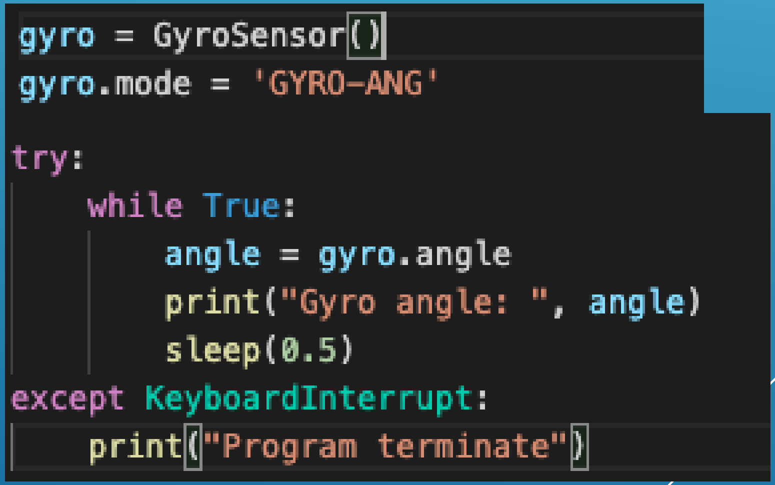


```python
us = UltrasonicSensor()
us.mode = 'US-DIST-CM'

try:
    while True:
        # 读取距离
        distance = us.distance_centimeters
        print("Distance: ", distance, "cm")
        sleep(0.5)
except KeyboardInterrupt:
    print("Program terminated")
```

GYRO SENSOR

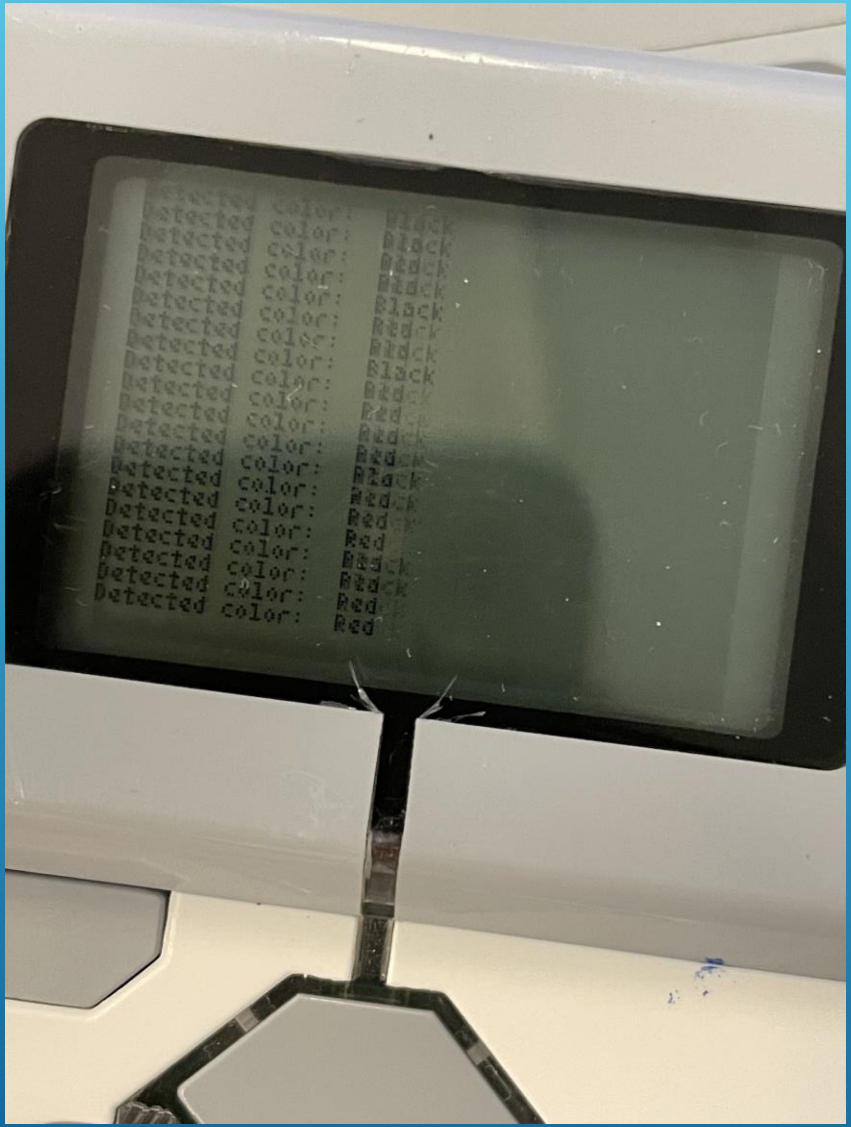


```
Gyro angle:   239
Gyro angle:   239
Gyro angle:   254
Gyro angle:   268
Gyro angle:   270
Gyro angle:   265
Gyro angle:   266
Gyro angle:   266
Gyro angle:   267
Gyro angle:   267
Gyro angle:   267
Gyro angle:   267
Gyro angle:   268
Gyro angle:   268
Gyro angle:   267
Gyro angle:   267
Gyro angle:   239
Gyro angle:   241
Gyro angle:   243
```

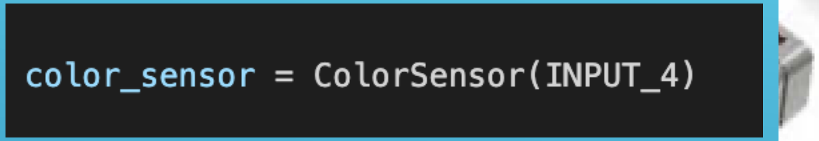


```
gyro = GyroSensor()
gyro.mode = 'GYRO-ANG'

try:
    while True:
        angle = gyro.angle
        print("Gyro angle: ", angle)
        sleep(0.5)
except KeyboardInterrupt:
    print("Program terminate")
```

COLOR SENSOR

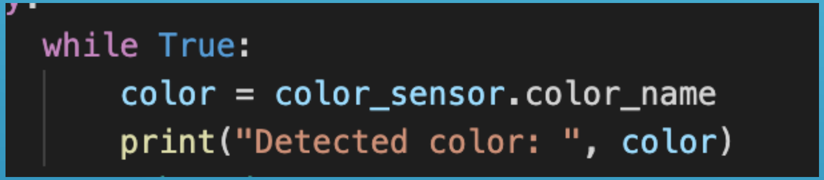


```
Detected color: Black
Detected color: Black
Detected color: Black
Detected color: Black
Detected color: Black
Detected color: Red
Detected color: Black
Detected color: Black
Detected color: Red
Detected color: Red
Detected color: Red
Detected color: Red
Detected color: Black
Detected color: Red
Detected color: Red
Detected color: Red
Detected color: Red
Detected color: Red
Detected color: Red
Detected color: Red
```

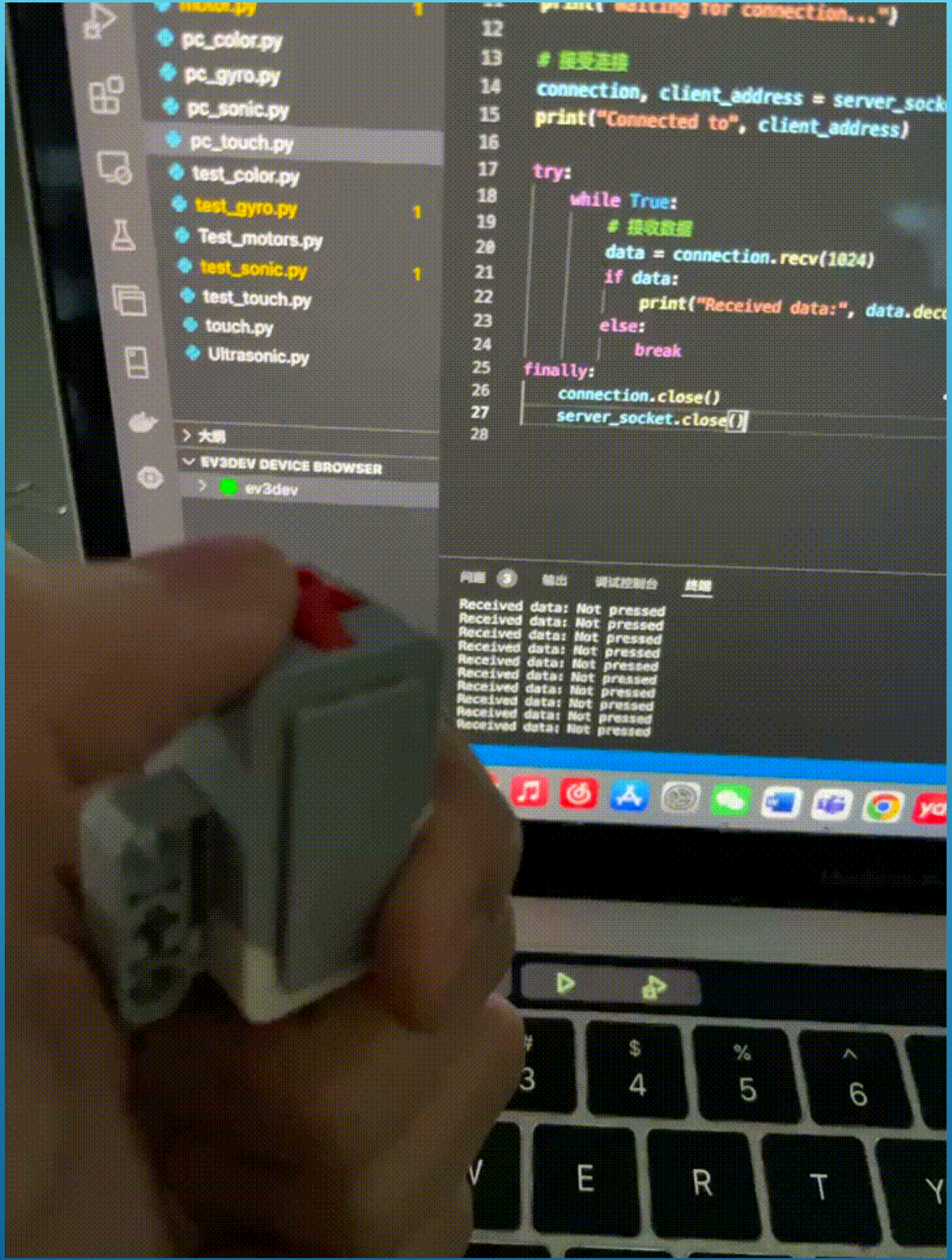


```python
color_sensor = ColorSensor(INPUT_4)
```

Color Sensor
- Recognizes seven different colors and measures light intensity.



```python
while True:
    color = color_sensor.color_name
    print("Detected color: ", color)
```

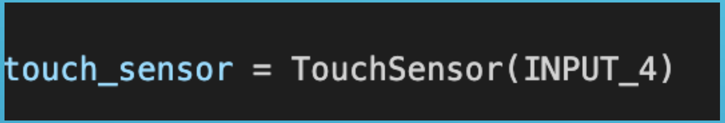


```
    print("Waiting for connection...")

    # 接受连接
    connection, client_address = server_sock
    print("Connected to", client_address)

try:
    while True:
        # 接收数据
        data = connection.recv(1024)
        if data:
            print("Received data:", data.deco
        else:
            break
finally:
    connection.close()
    server_socket.close()
```

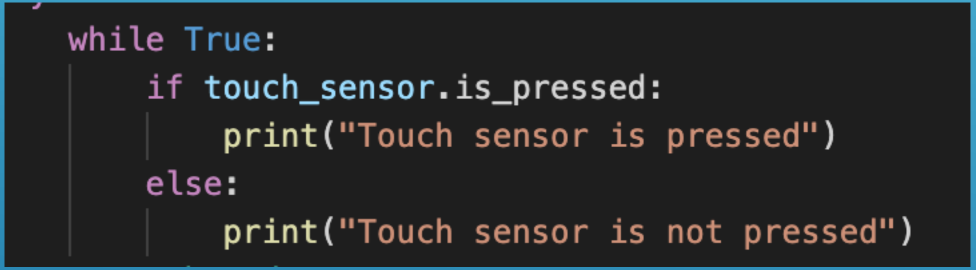
```
Received data: Not pressed
Received data: Not pressed
Received data: Not pressed
Received data: Not pressed
Received data: Not pressed
Received data: Not pressed
Received data: Not pressed
Received data: Not pressed
Received data: Not pressed
Received data: Not pressed
```

TOUCH SENSOR

Touch Sensor
- Recognizes three conditions—touched, bumped, and released.



```
touch_sensor = TouchSensor(INPUT_4)
```



```
    while True:
        if touch_sensor.is_pressed:
            print("Touch sensor is pressed")
        else:
            print("Touch sensor is not pressed")
```

FUTURES STEPS

Build the robot

Sensors Integration
Structure Building
Program implantation

Achieve the bronze step

Movement
Target Sensing
Tracking Position
Communication
Calibrating