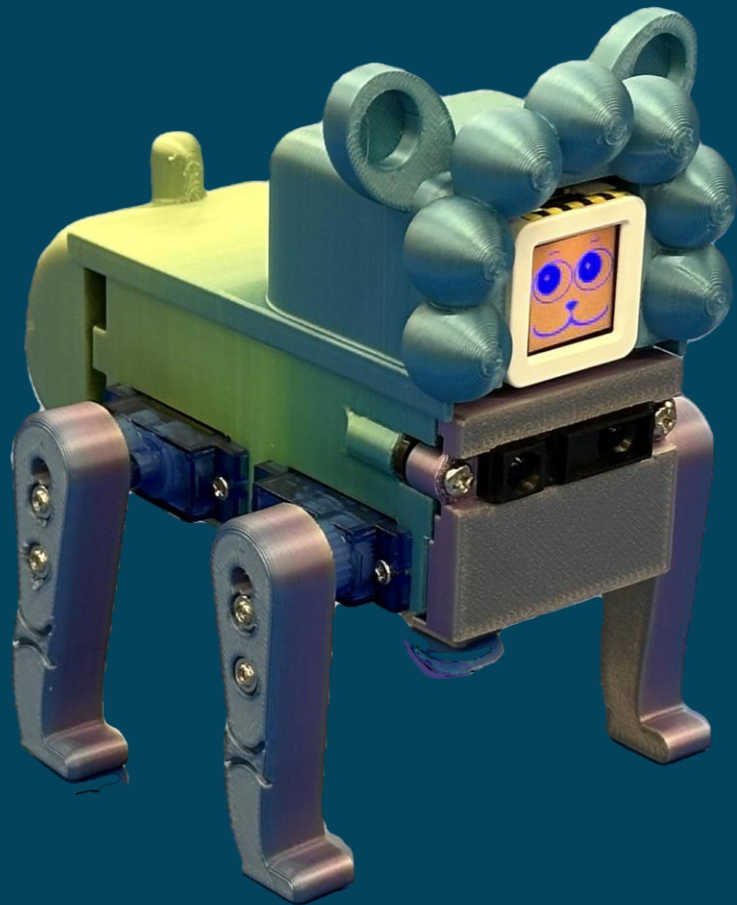


Workshop on Designing and Assembling a Walking Robot 設計與組 裝可步行機械人工作坊



STEM Education Centre
26 September 2024

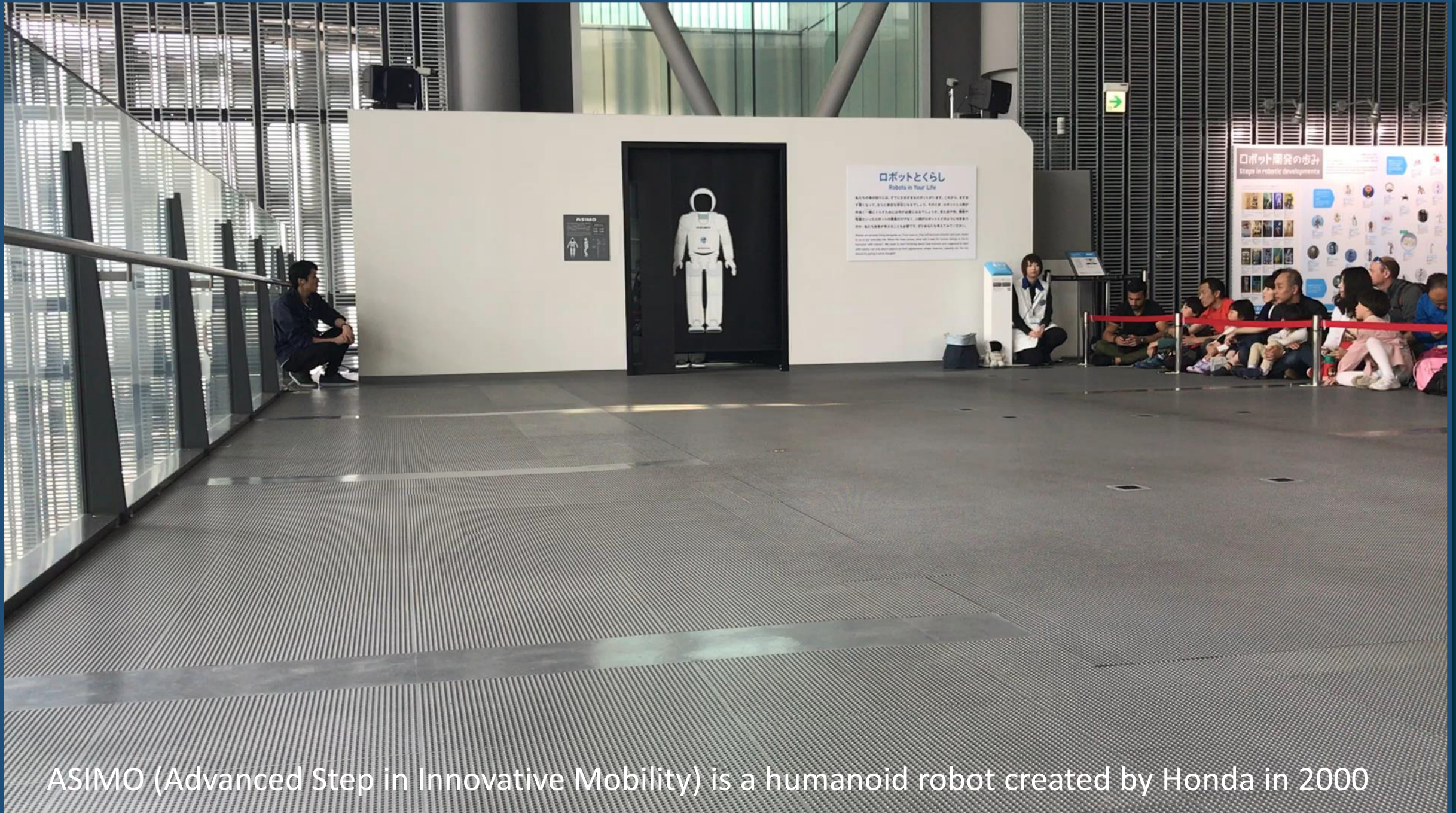
Workshop on Designing and Assembling a Walking Robot 設計與組裝可步行機械人工作坊

Time 時間	Content/Activity 內容/活動
14:30 – 14:45	Introduction to the design of legged robots and related learning and teaching resource materials 簡介多足機械人的設計及相關學與教資源
14:45 – 15:15	Introduction to the programmable control of robots (e.g. sensors, servo motors, microcontrollers) and related components of building a walking robot 簡介機械人的可編程控制 (例如感應器、伺服電動機、微控制器) 及建構步行機械人的相關元件
15:15 – 16:15	Hands-on activities: <ul style="list-style-type: none">• Design and assembly of the robot• Wiring of the robot 動手做活動: <ul style="list-style-type: none">• 機械人的設計與組裝• 機械人的接線
16:15 – 16:50	Demonstration and explanation of how the robots with uploaded codes work 演示及解釋已上載代碼的機械人如何運作
16:50 – 17:00	Q&A 問與答

Aims of the workshop today

- We think that proficiency and interest for STEAM can be enhanced via robotics.
- Robotics activities are a great way to get students engaged and familiarised with STEAM.
- Many students, when they think of robotics, associate them with movies, TV shows, toys, video games and other exciting things that they do in their fun time.
- This enthusiasm may generate interest in a future STEAM career or help students feel more comfortable in STEAM projects

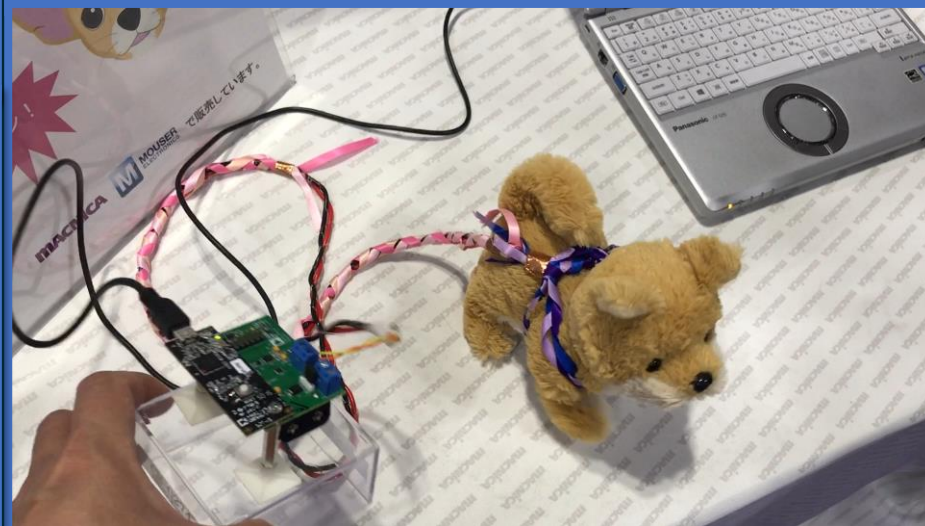
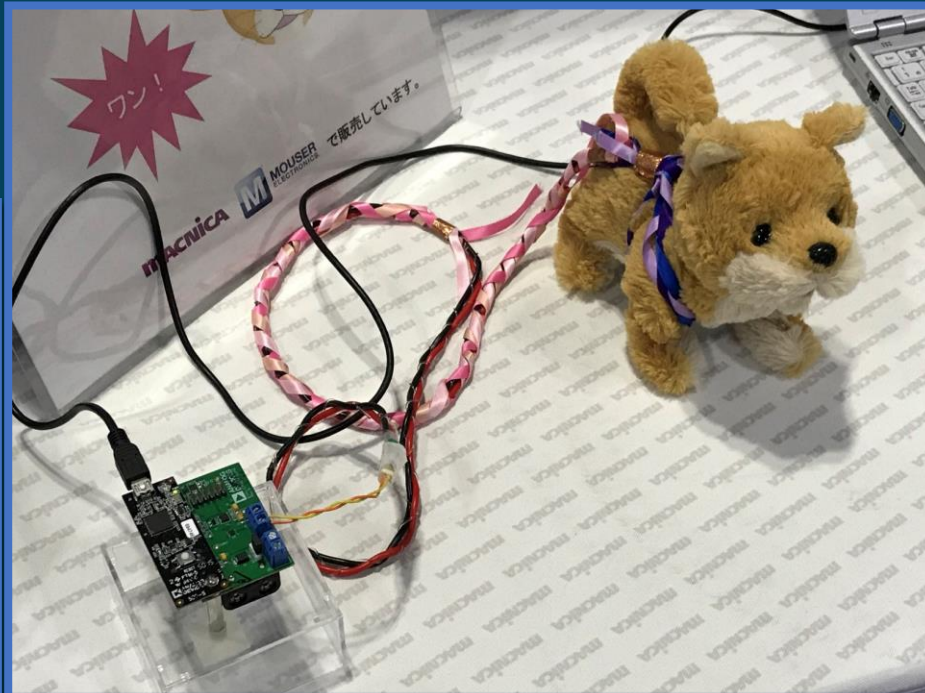
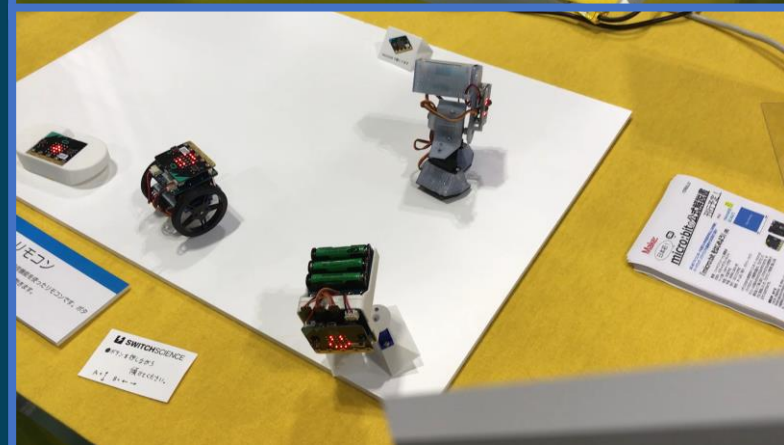
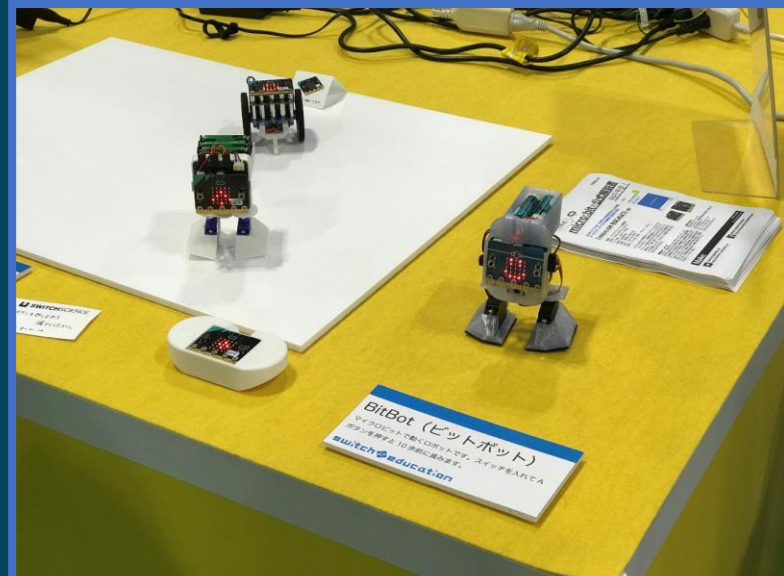
Robotics Can Improve Proficiency and Enthusiasm for STEAM



ASIMO (Advanced Step in Innovative Mobility) is a humanoid robot created by Honda in 2000

Legged robot vs wheeled robot

Maker Faire 2017 Tokyo



Bipedal (2-legged) robots vs Quadrupedal (4-legged) robots



Examples of robotics use in Education

- **Children with autism** are learning communication and social skills
- Students with **developmental issues** and attention disorders are learning focus.



Department of Educational Psychology
The Chinese University of Hong Kong

[HOME](#) [ABOUT US](#) [PEOPLE](#) [RESEARCH](#) [TEACHING](#) [SERVICE](#)

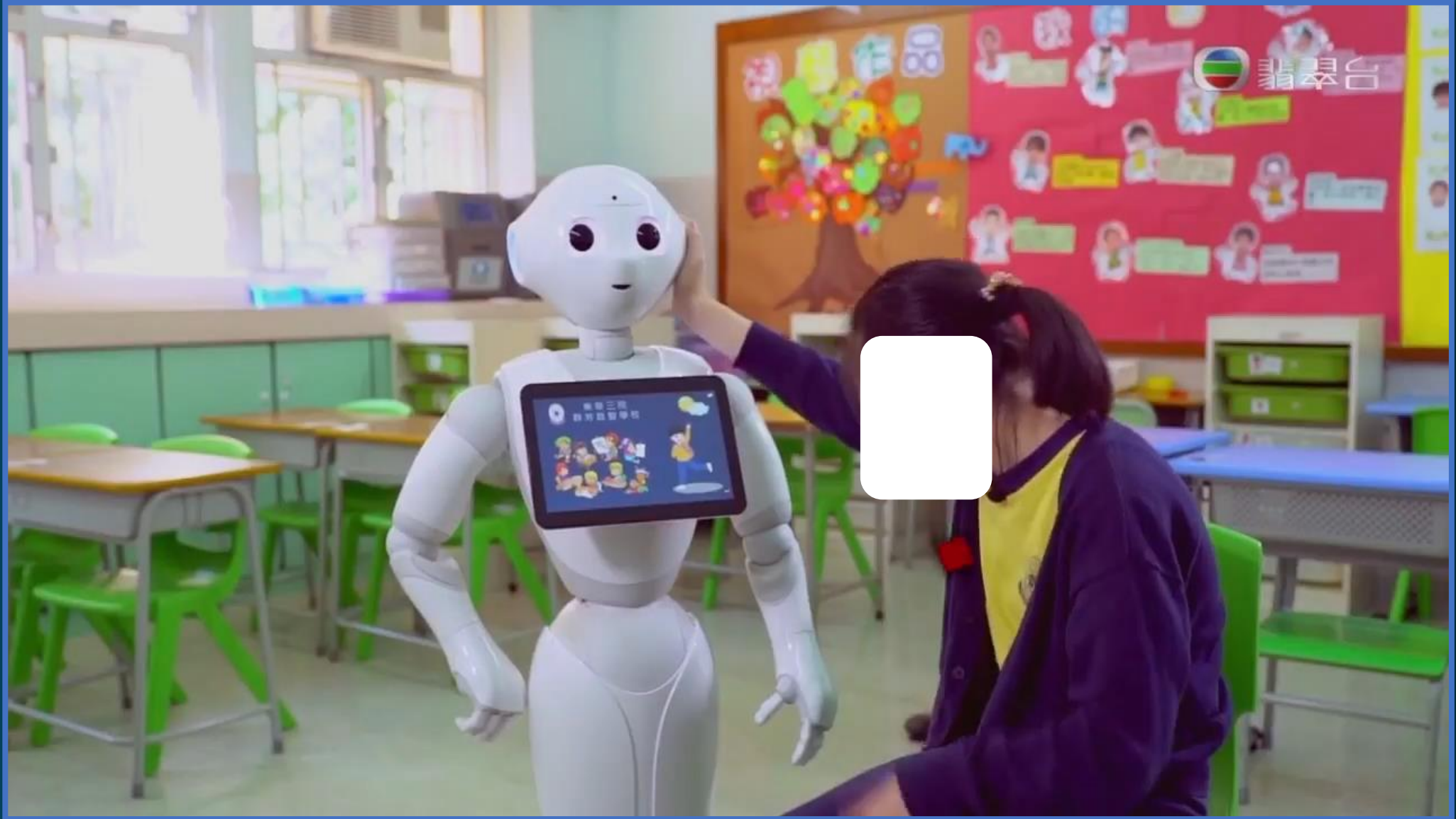
Our hands speak: Robot-based intervention to promote gestural communication in children with Autism Spectrum Disorders (ASD)

妙手可言：使用社交機器人支援自閉症兒童手勢溝通計劃——用手告訴您
[A project led by Prof. So Wing-chee, Catherine]

Gesture, spontaneous hand movement produced with or without speech, is an effective channel for communication. Our previous studies are the first showing school-aged children with Autism Spectrum Disorders (ASD) are found to have gesture delay during their middle and late childhood (So, Lui, Wong, & Sit, 2015a; So, Wong, Lui, & Yip, 2015b; So & Wong, 2016). The difficulty with gesture use is more salient among those with weaker social and communication skills (So, Wong, & Lam, 2016a). Collaborating with Faculty of Engineering, Faculty of Arts, and Faculty of Medicine, our team thus design and implement home-based and school-based intervention programs to promote their gestural communication skills using humanoid social robot. We collaborate with a number of organizations that serve children with ASD in Hong Kong including Hong Chi Association, SAHK, Hong Kong Christian Service, STEP Center for Child Development, and Hong Kong Sheng Kung Hui. Our programs are funded by the CUHK Knowledge Transfer Fund, Quality Education Fund, and Social Innovation Enterprise Fund.



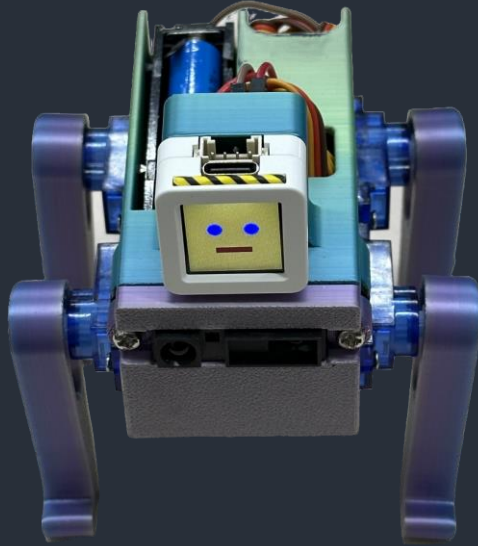
Use robots to teach student with Intellectual disability



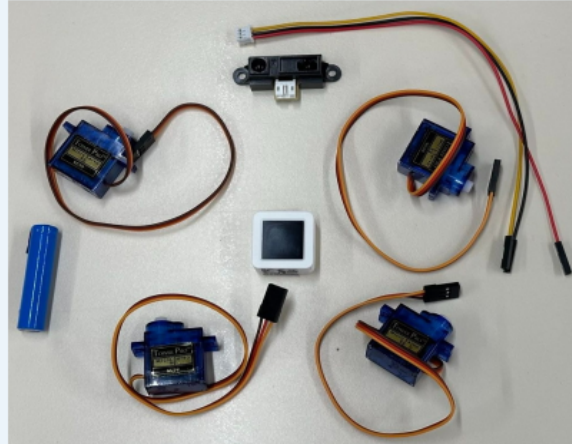
<https://www.youtube.com/watch?v=2IYHkGONIkQ>

It is possible
to make robot
programming
simple and
engaging

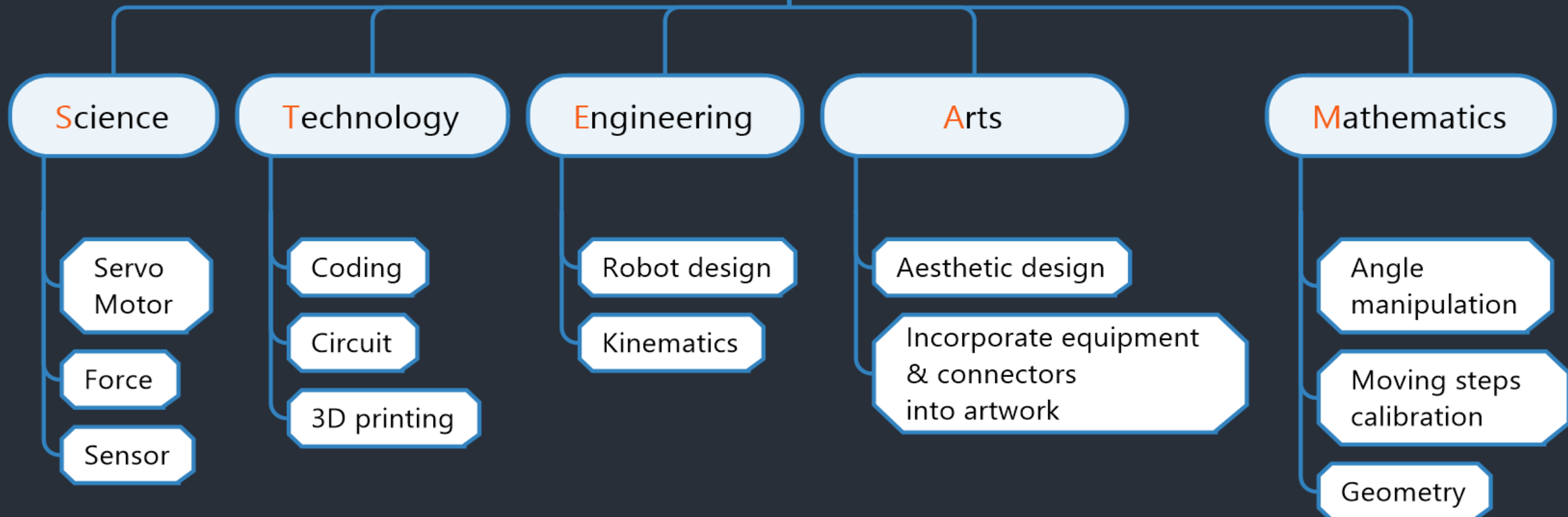





Today's STEAM Project



Robotics is an interdisciplinary subject that combines hardware, software, algorithm and control.



Related learning and teaching resource materials

**Education Bureau**
The Government of the Hong Kong Special Administrative Region
of the People's Republic of China




[Home](#) [Text Size](#) [Search](#)

[Latest News](#) | [About EDB](#) | [Press Release](#) | [Education System and Policy](#) | [Curriculum Development and Support](#) | [Students and Parents Related](#) | [Teachers Related](#) | [School Administration and Management](#) | [Public and Administration Related](#) | [Access to Information](#) | [Contact Us](#)

[Home](#) > [Curriculum Development and Support](#) > [Key Learning Areas](#) > [Technology Education](#) > [Technology Education – References & Resources](#) > [Technological Subjects](#)

S1-3 STEM Education – Learning and Teaching Resources

Project Learning learning and teaching resources

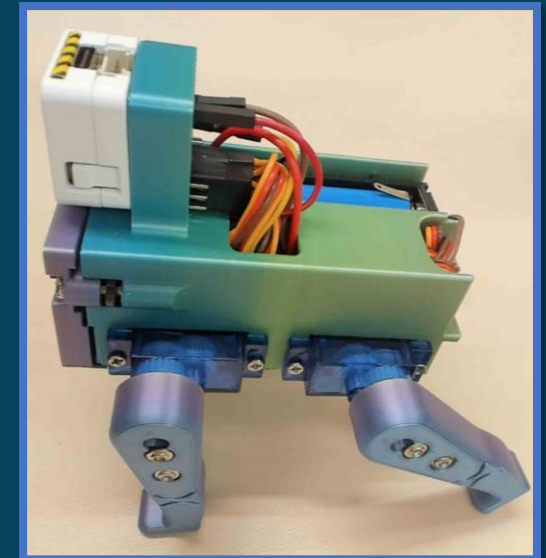
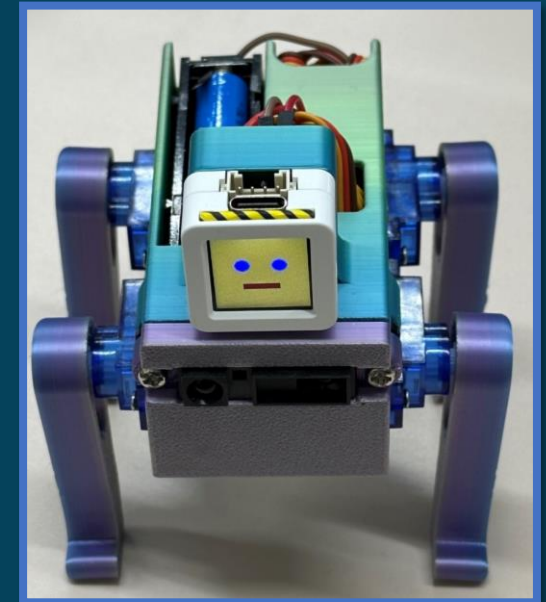
Topics	English Version	Chinese Version	Remarks
Robotics Project Learning	Details 	Details 	Download programme code 

Robotics learning and teaching resources

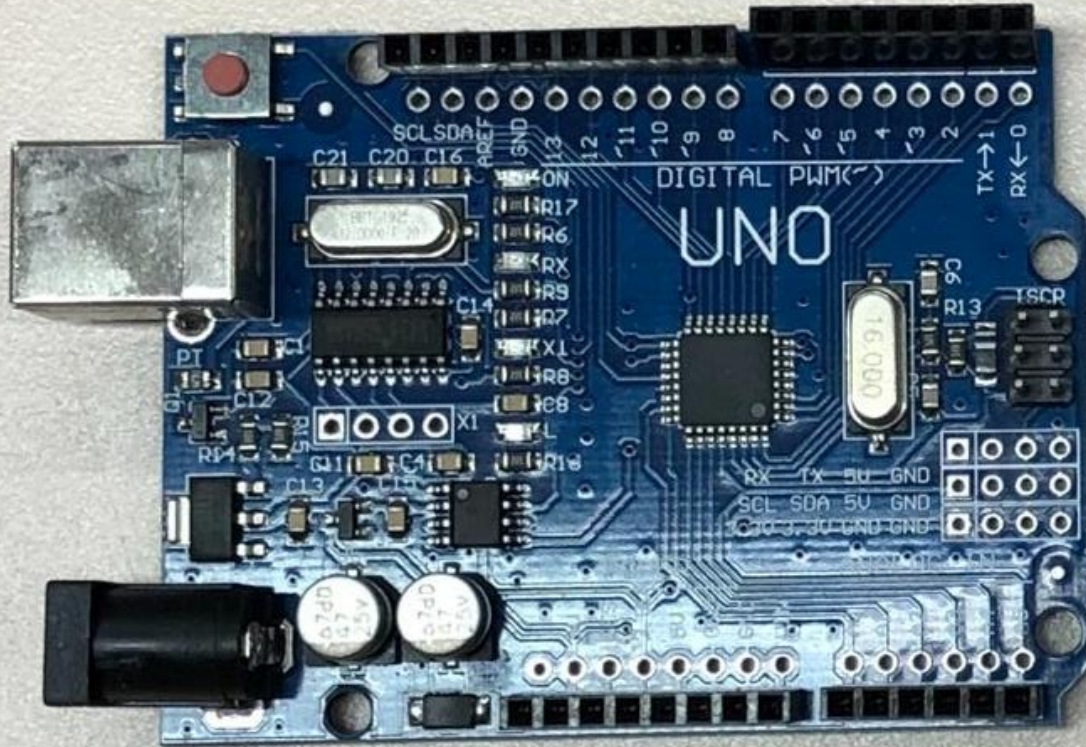
Level	Topics	English Version	Chinese Version	Remarks
Basic Unit 1	Basic control of electronic components	Details	Details	Download programme code
Basic Unit 2	Input and output in robotics	Details	Details	Download programme code

Considerations in designing the robot

- Have **legs** and able to **walk**
- Have a **display** that shows its face
- Reduce the body to the **smallest possible size**
- Able to operate on **small battery**
- Use the **least amount of equipment** possible
- Electrical wires need to be **housed neatly** within designed ducts
- **Most students** can do it



Micro-controller unit for the robot



Arduino (Uno)


M5 AtomS3




Micro:bit

Development of new hardware → Opportunities for innovation


Choose the Micro-controller unit for this Project



ESP32-S3FN8
WiFi / DualCore
8M-FLASH
0.85" IPS-LCD
128 x 128 P
6-DoF IMU
MPU6886
Button x1
GPIOs x 6
IR LED x1
USB CDC
PortA x1



ESPRESSIF

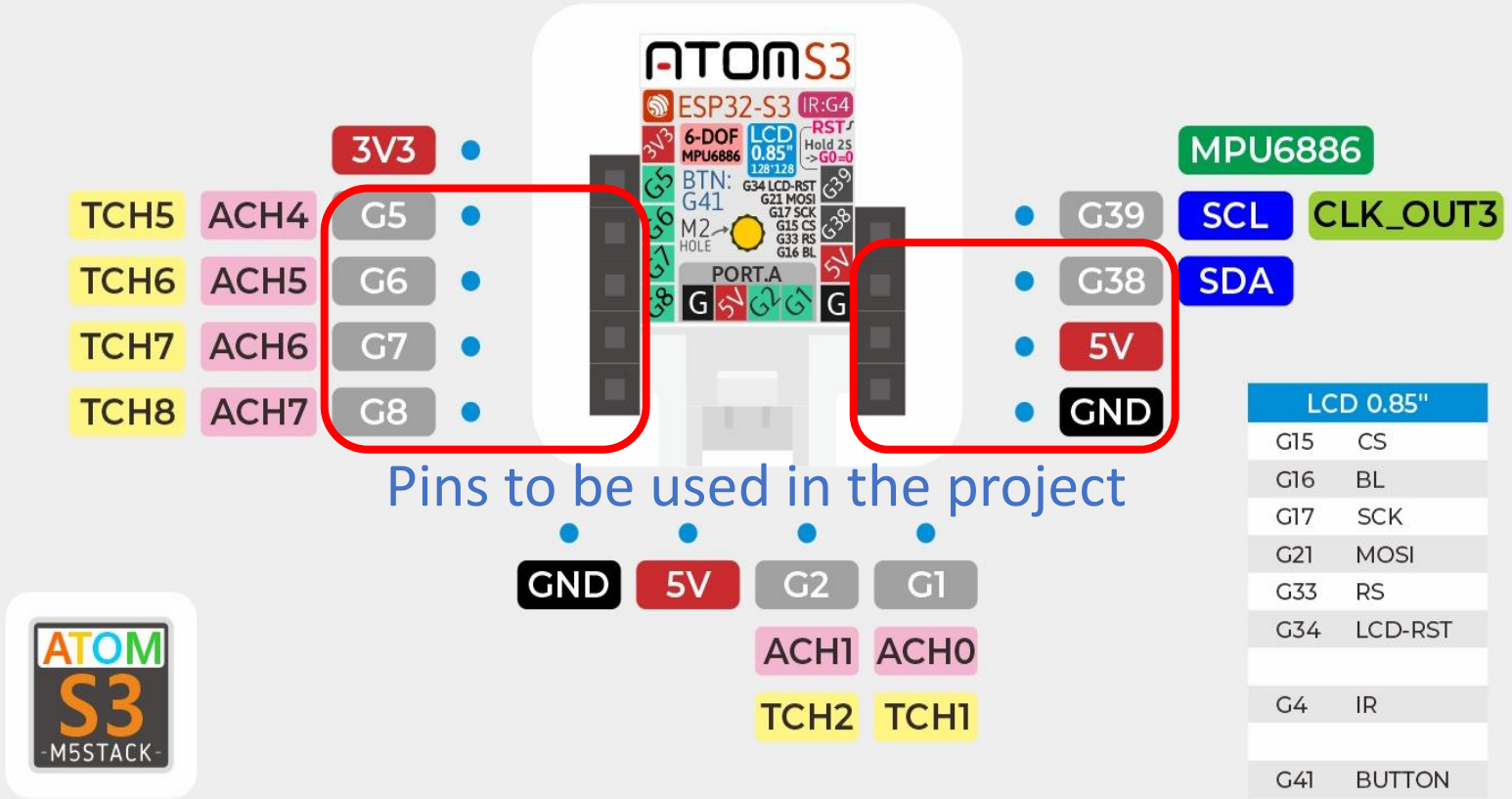


ATOM S3
-M5STACK-

Legend:

- CLK OUT
- I2C
- ADC**
- TOUCH
- GPIO
- POWER
- GND

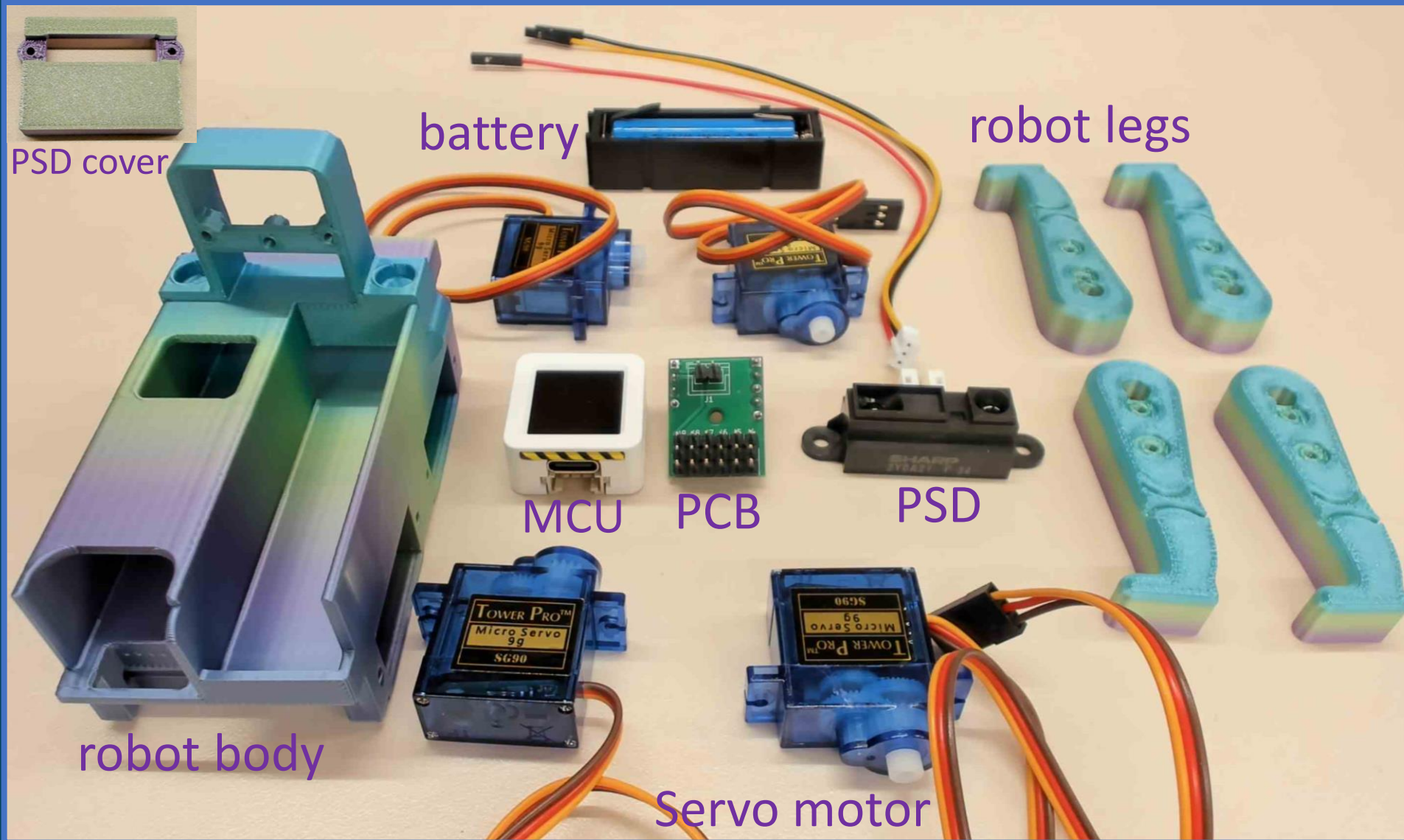
Pins to be used in the project



The diagram shows the ATOM S3 M5STACK module with various pins highlighted. A red box highlights the 3V3, G5, G6, G7, and G8 pins. Another red box highlights the G39, G38, 5V, and GND pins. A third red box highlights the GND, 5V, G2, and G1 pins. A fourth red box highlights the ACH1, ACH0, TCH2, and TCH1 pins. A legend at the bottom shows color-coded boxes for CLK OUT (green), I2C (blue), ADC (pink), TOUCH (yellow), GPIO (grey), POWER (red), and GND (black).

LCD 0.85"	
G15	CS
G16	BL
G17	SCK
G21	MOSI
G33	RS
G34	LCD-RST
G4	IR
G41	BUTTON

Other components of the robot



MCU:
Micro-controller unit

PCB:
Printed circuit board

PSD:
Position sensitive
detector

Coding to operate the robot

```
#include <M5AtomS3.h>
#include <DabbleESP32.h>

#define Default_SETTINGS

const uint8_t Srv0 = 6; //GPIO pin for Right Front leg servo
const uint8_t Srv1 = 7; //GPIO pin for Right Back leg servo
const uint8_t Srv2 = 8; //GPIO pin for Left Front leg servo
const uint8_t Srv3 = 38; //GPIO pin for Left Back leg servo
const uint8_t Adc0 = 5; //Analog to Digital converter port

const uint8_t srv_CH0 = 0, srv_CH1 = 1, srv_CH2 = 2, srv_CH3 = 3; // define servo channels
const double PWM_Hz = 50; // Pulse width modulation frequency => PWM period = 1/50 = 20ms
const uint8_t PWM_resolution = 14; // PWM resolution 14bit (0~16383)

int pulse_All_Left = 409; // 0deg : 2^14(bit) x 0.5ms (pulse) / 20ms (PWM period)
int pulse_All_Right = 2048; //180deg: 2^14(bit) x 2.5ms (pulse) / 20ms (PWM period)

int all_left = 0;
int all_right = 180;

int HomePosAng[] = {85,95,103,93}; // initial servo angle
int ang0[4]; // temporary variable
int ang1[4]; // temporary variable
float ts=120; // move on to the next step after 120ms
float td=10; // control the speed of the servo angle changes
```

```
void setup() {
  Serial.begin(151200);
  M5.begin(true, true, false, false); // Initialise M5AtomS3 (LCD, USB serial, I2C(38,39), LED)
  M5.Lcd.setRotation(4); // Rotate the screen 90 deg in counterclockwise dir
  Dabble.begin("STEMcentre-robot-1"); // set bluetooth name for connection

  pinMode(Srv0, OUTPUT);
  pinMode(Srv1, OUTPUT);
  pinMode(Srv2, OUTPUT);
  pinMode(Srv3, OUTPUT);
  pinMode(Adc0, INPUT);

  // configure servo PWM channel and frequency
  ledcSetup(srv_CH0, PWM_Hz, PWM_resolution);
  ledcSetup(srv_CH1, PWM_Hz, PWM_resolution);
  ledcSetup(srv_CH2, PWM_Hz, PWM_resolution);
  ledcSetup(srv_CH3, PWM_Hz, PWM_resolution);

  // Configure servo pins and channels
  ledcAttachPin(Srv0, srv_CH0);
  ledcAttachPin(Srv1, srv_CH1);
  ledcAttachPin(Srv2, srv_CH2);
  ledcAttachPin(Srv3, srv_CH3);

  face_normal();

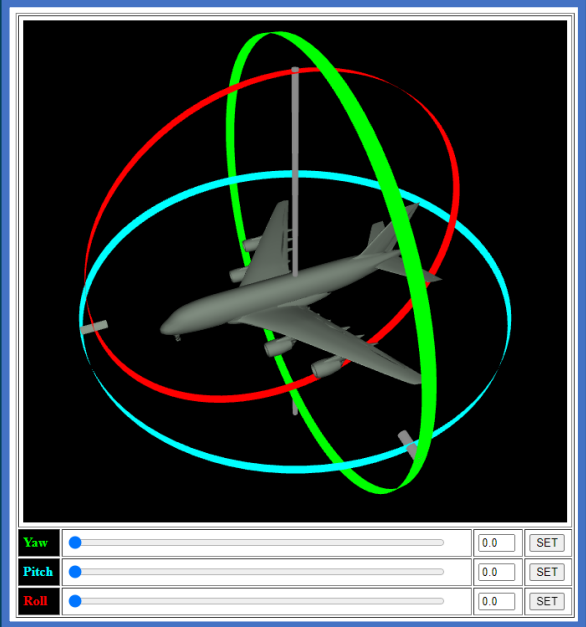
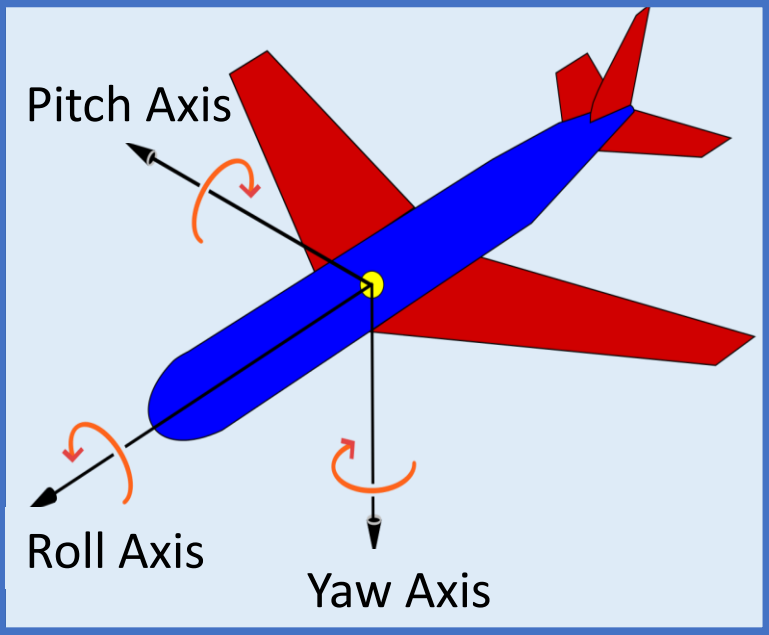
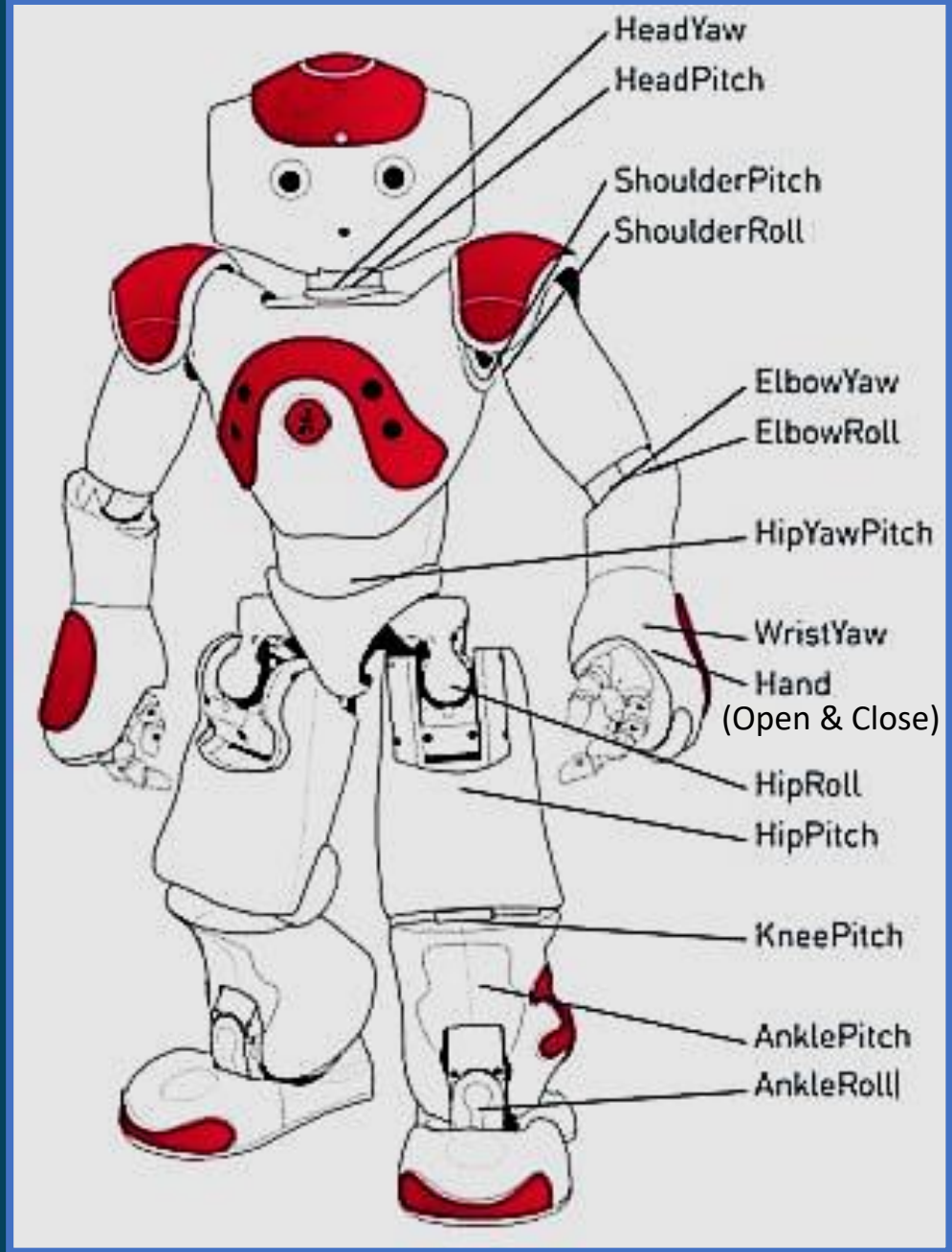
  Initial_setting();
}
```



Degrees of freedom in a robot

To impose a certain kind of motion to happen between two rigid parts in a single degree of freedom, use a **servo motor**.

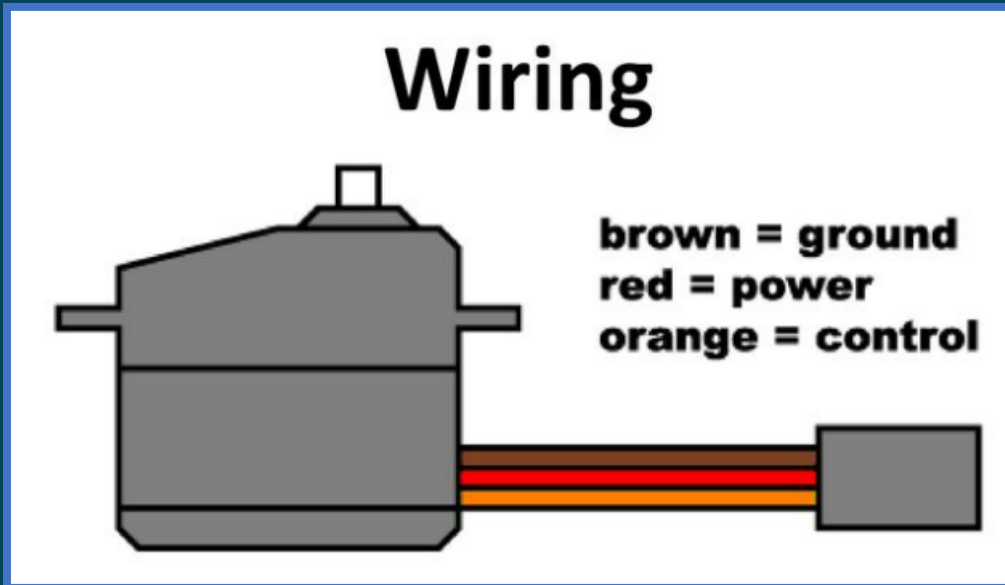
Body Part	Degrees of Freedom (可動關節)
Neck	2
Shoulders & Arms	6 x 2
Hip & Legs	5 x 2 + 1
Total	25



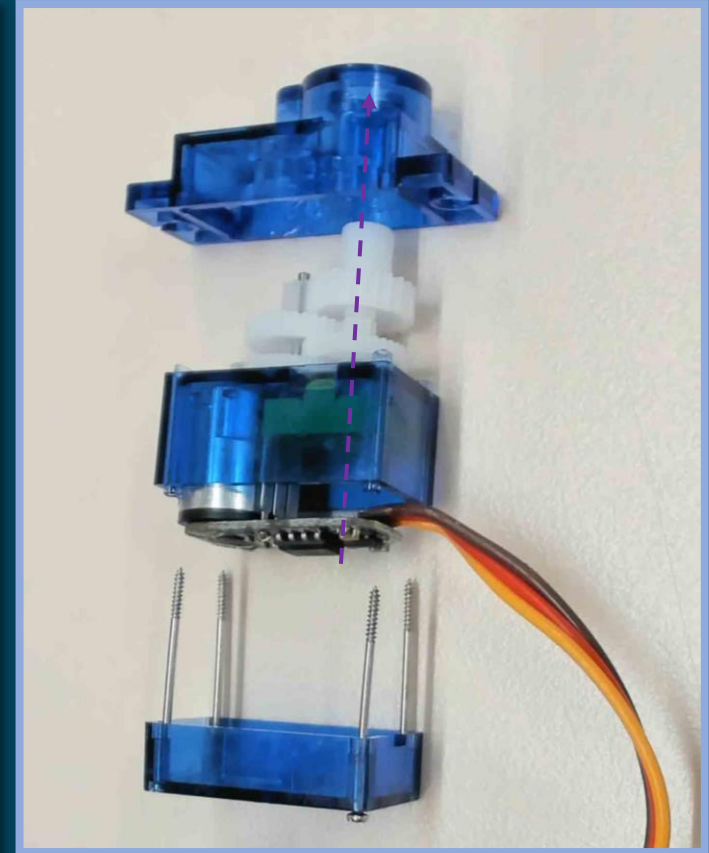
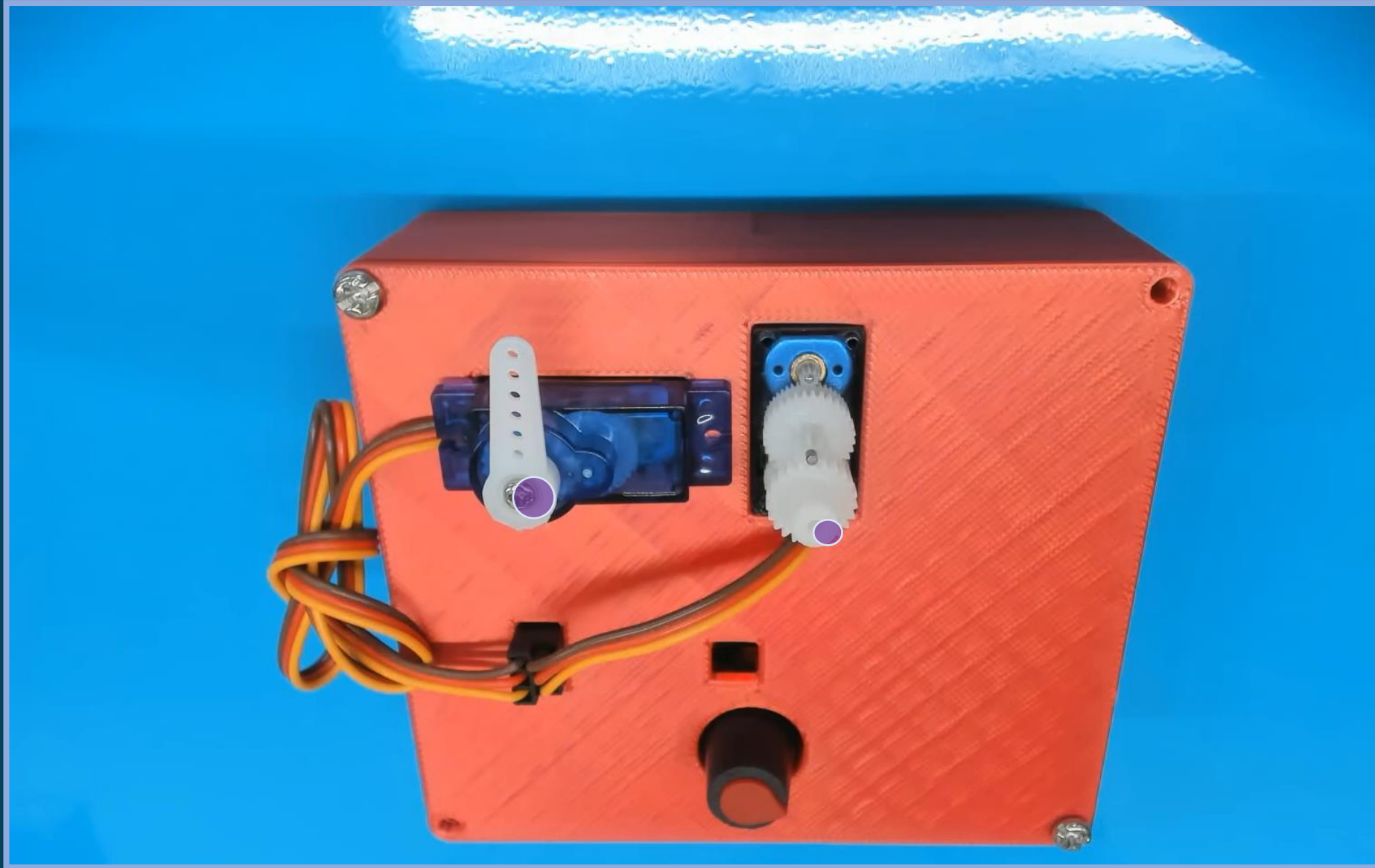
What is a Servo motor ?

Servo motors consist of:

1. Gear reduction (slow but strong)
2. Position sensor
3. Electronic circuitry

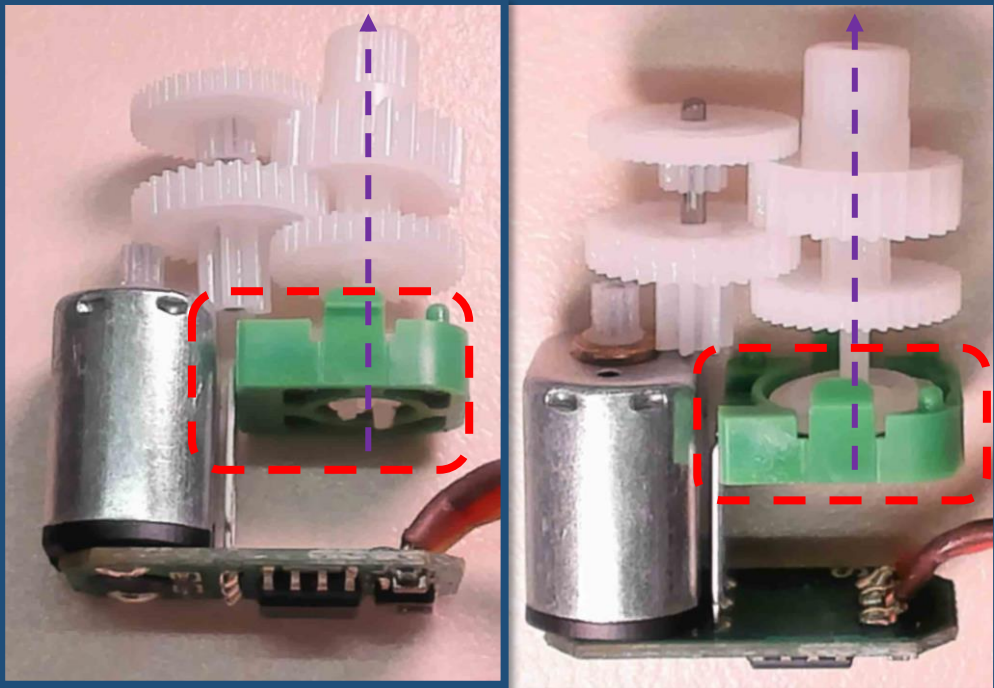


How Does the SG90 Servo Motor Works

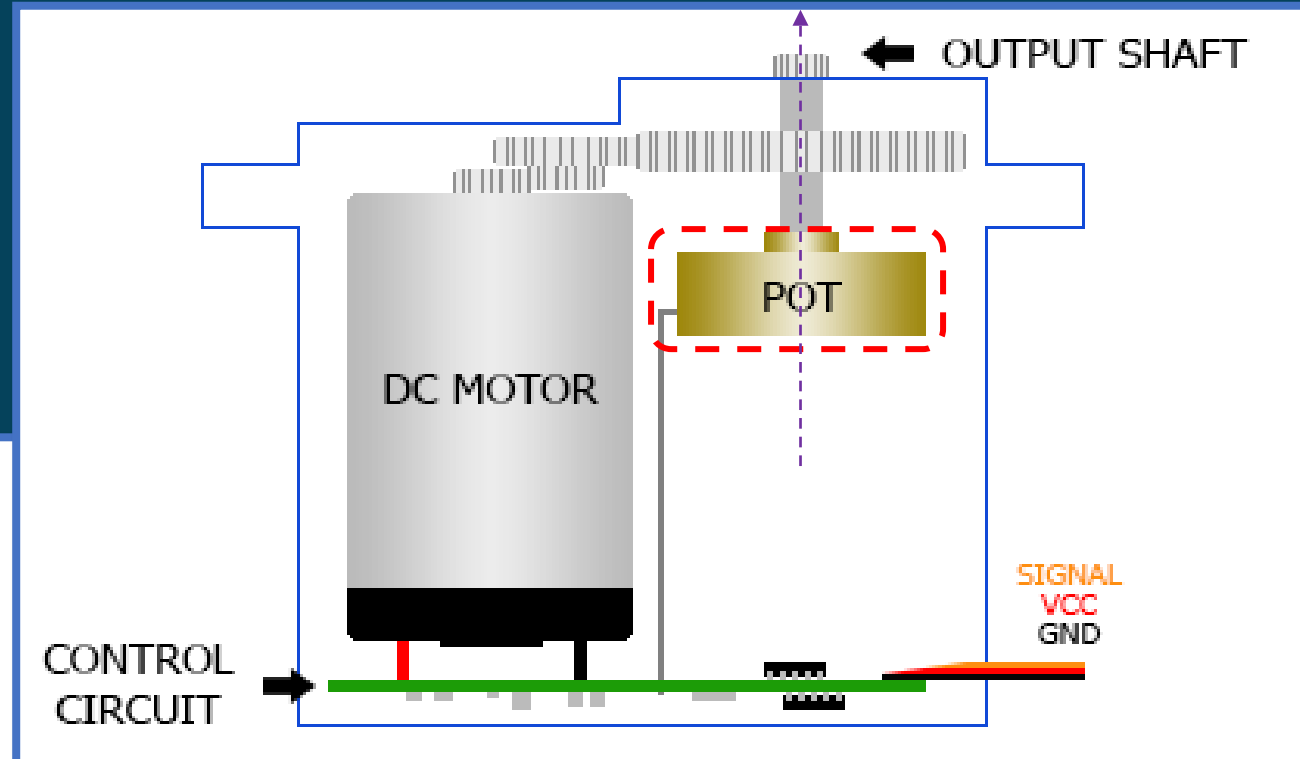
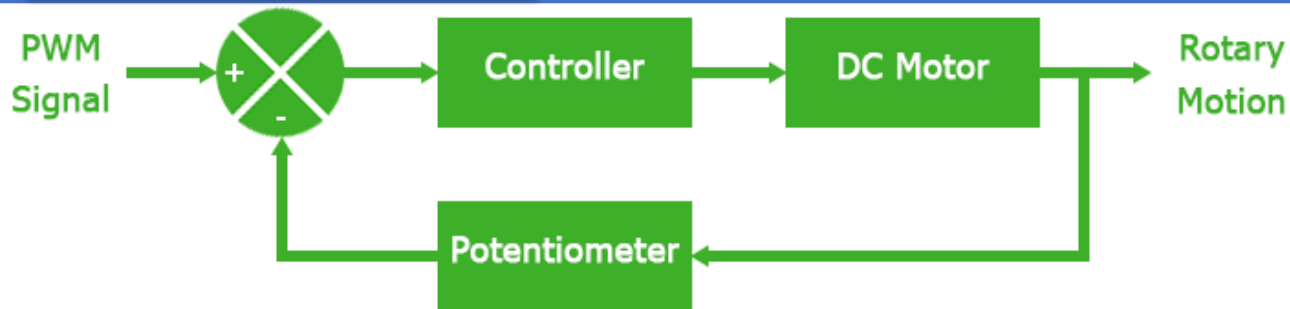


How does a Servo motor work ?

A servo motor has an electronic board that accepts **PWM (Pulse Width Modulation 脈寬調變)** signals and measures its on-time pulse width. The servo motor also has a potentiometer that helps in keeping track of the shaft position.

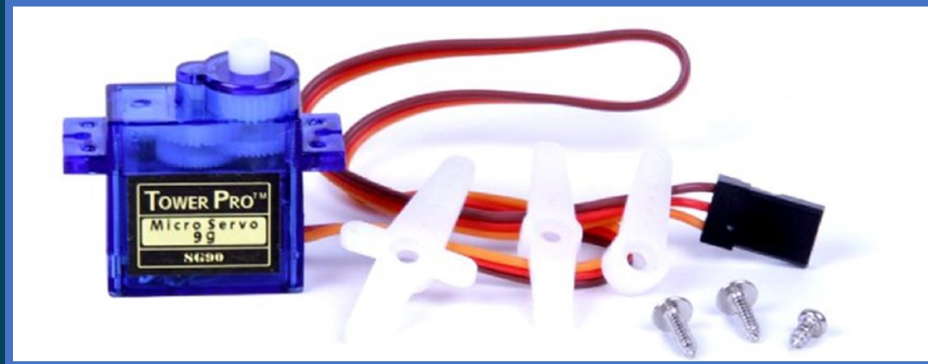


The embedded board continuously detects and corrects the unintended shift in the shaft position. The target position is maintained by continuous error correction between the shaft position and the user input.

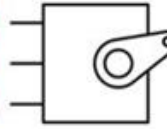


How can we use PWM signals to operate a servo motor?

To control the motion of the SG90 servo motor we need to apply a **PWM** signal to the orange wire to achieve the desired position.



PWM=Orange (⏏)
Vcc=Red (+)
Ground=Brown (-)



PWM period = 20ms

The duration of "on-time" is called the pulse width.

Set PWM period = 20ms (frequency 50Hz)
工作週期 工作頻率

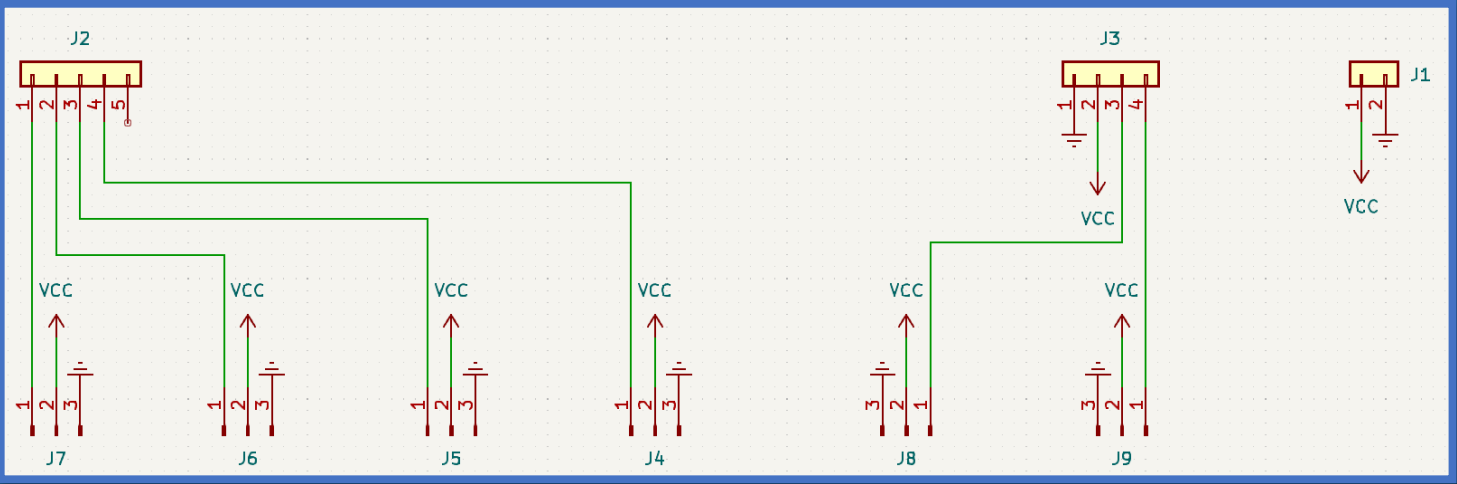
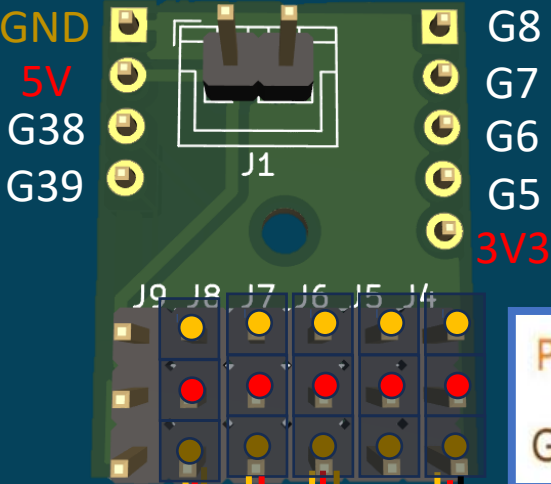
0.5ms "on-time" pulse width corresponds to 0°



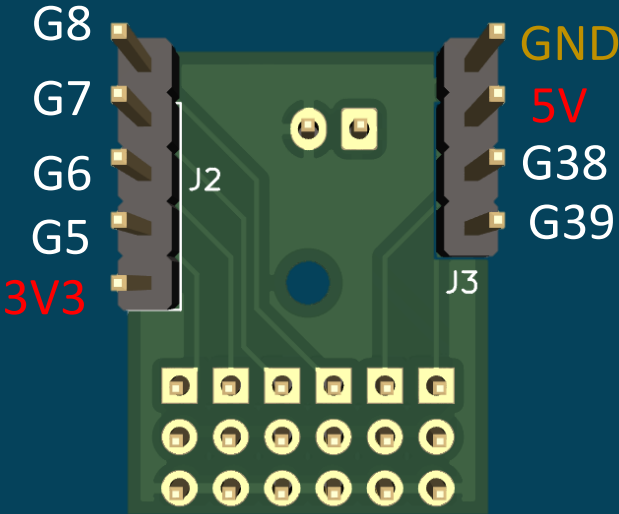
2.5ms "on-time" pulse width corresponds to 180°

Simple Wiring Diagram

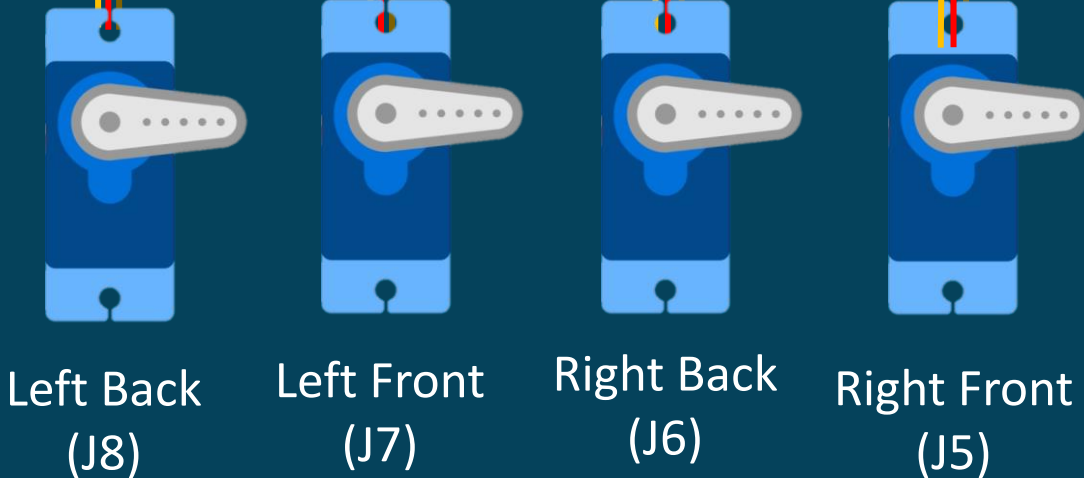
Back side of the PCB



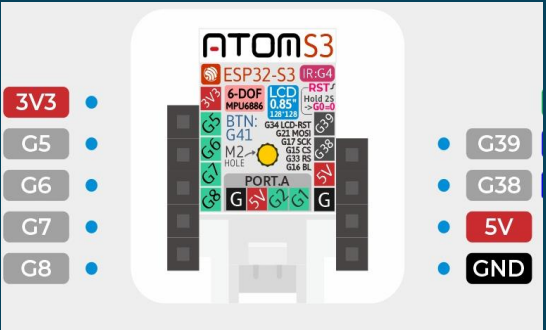
Front side of the PCB



Servos

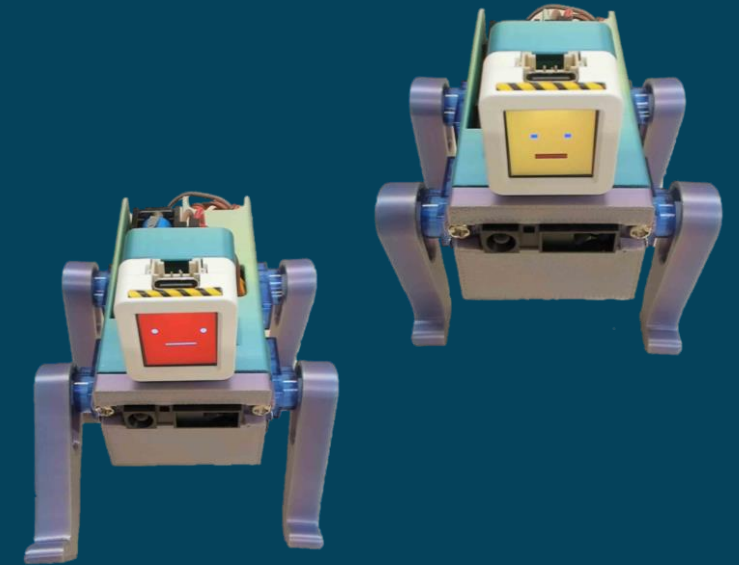


Position sensitive detector (J4)



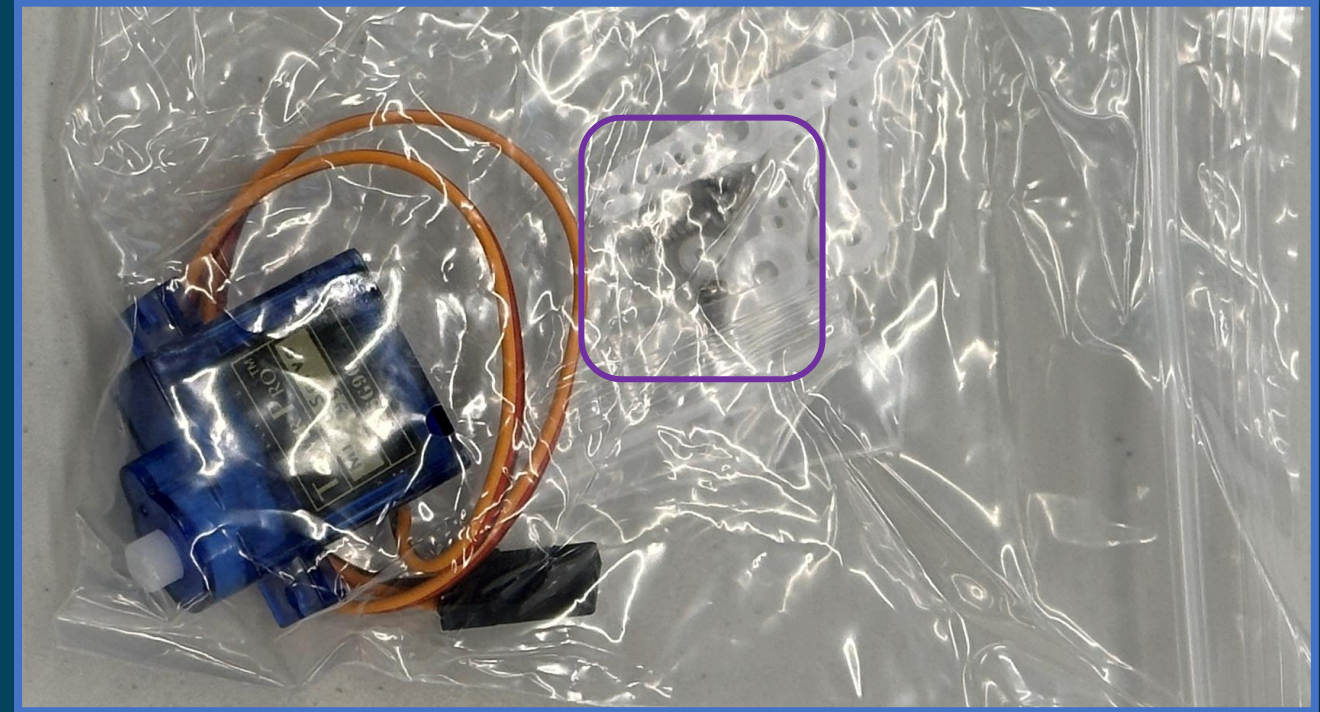
Features that the robot must have

- Able to move forward, backward, turn left or right
- Can adopt a few poses
- Facial content alterations are possible
- Able to be operated remotely via smartphone apps like **Dabble**
- Able to regulate its movement in response to the distance of an object in front of it



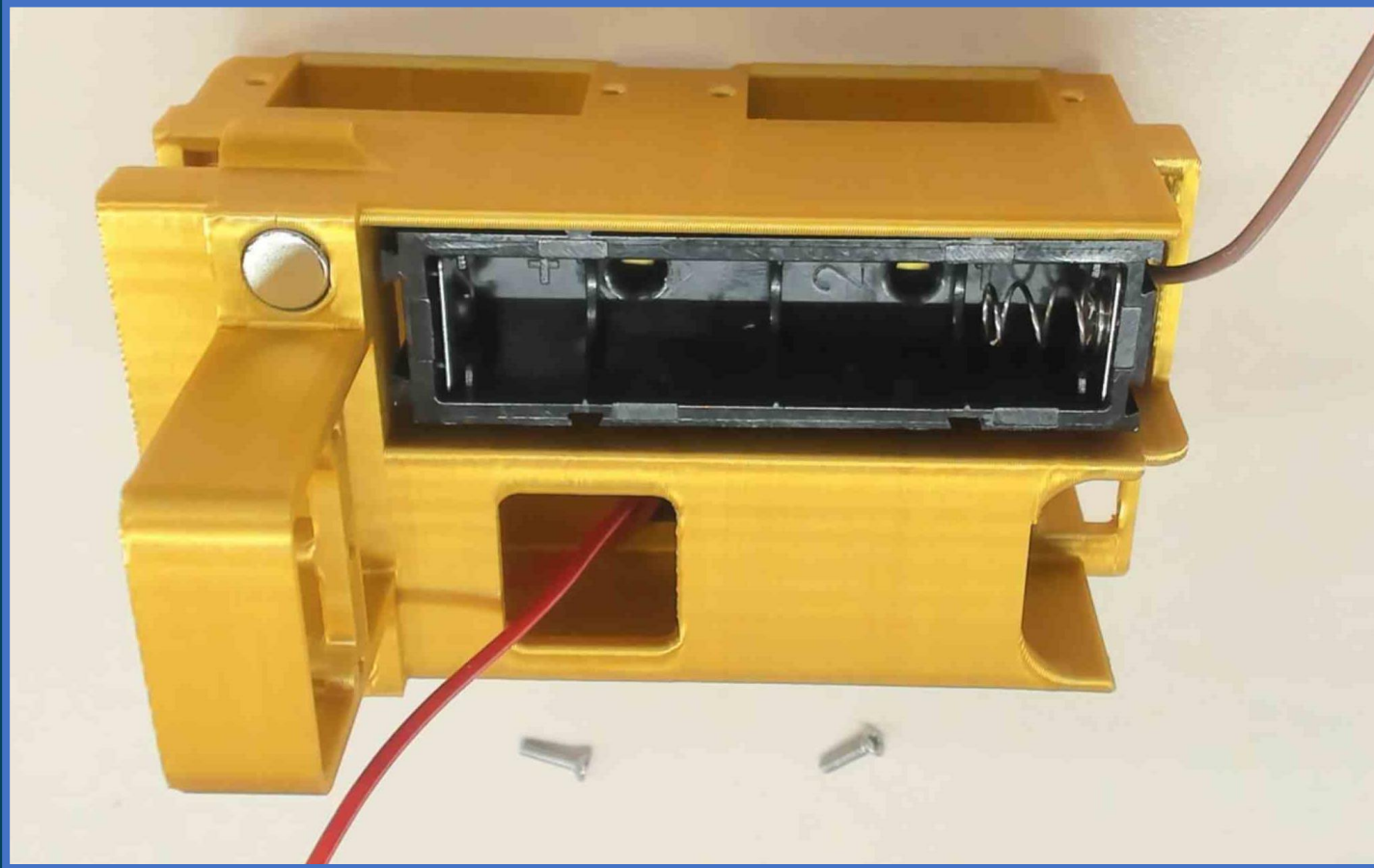
Dabble

Assembly of the robot:



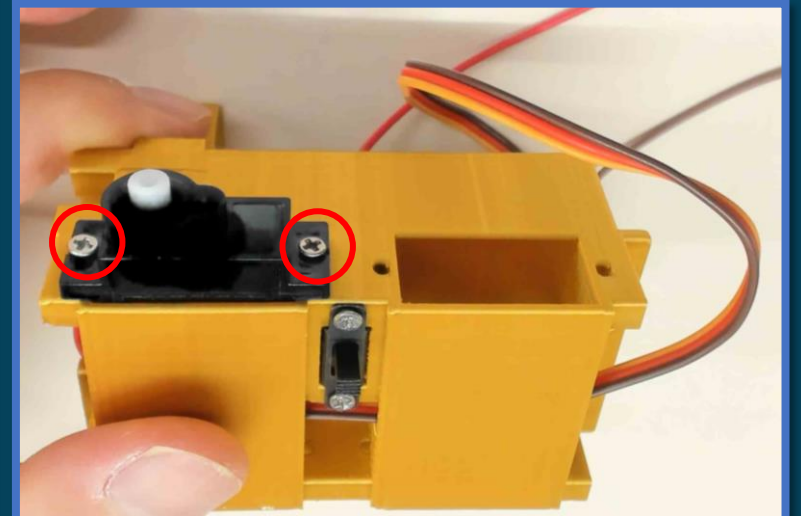
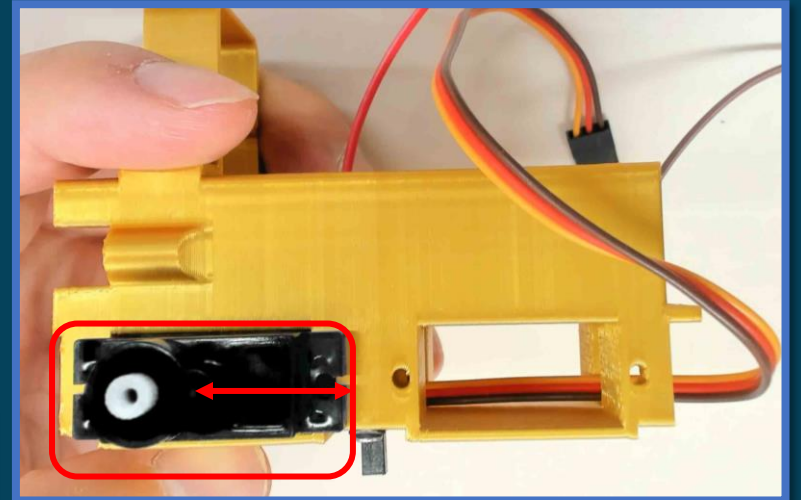
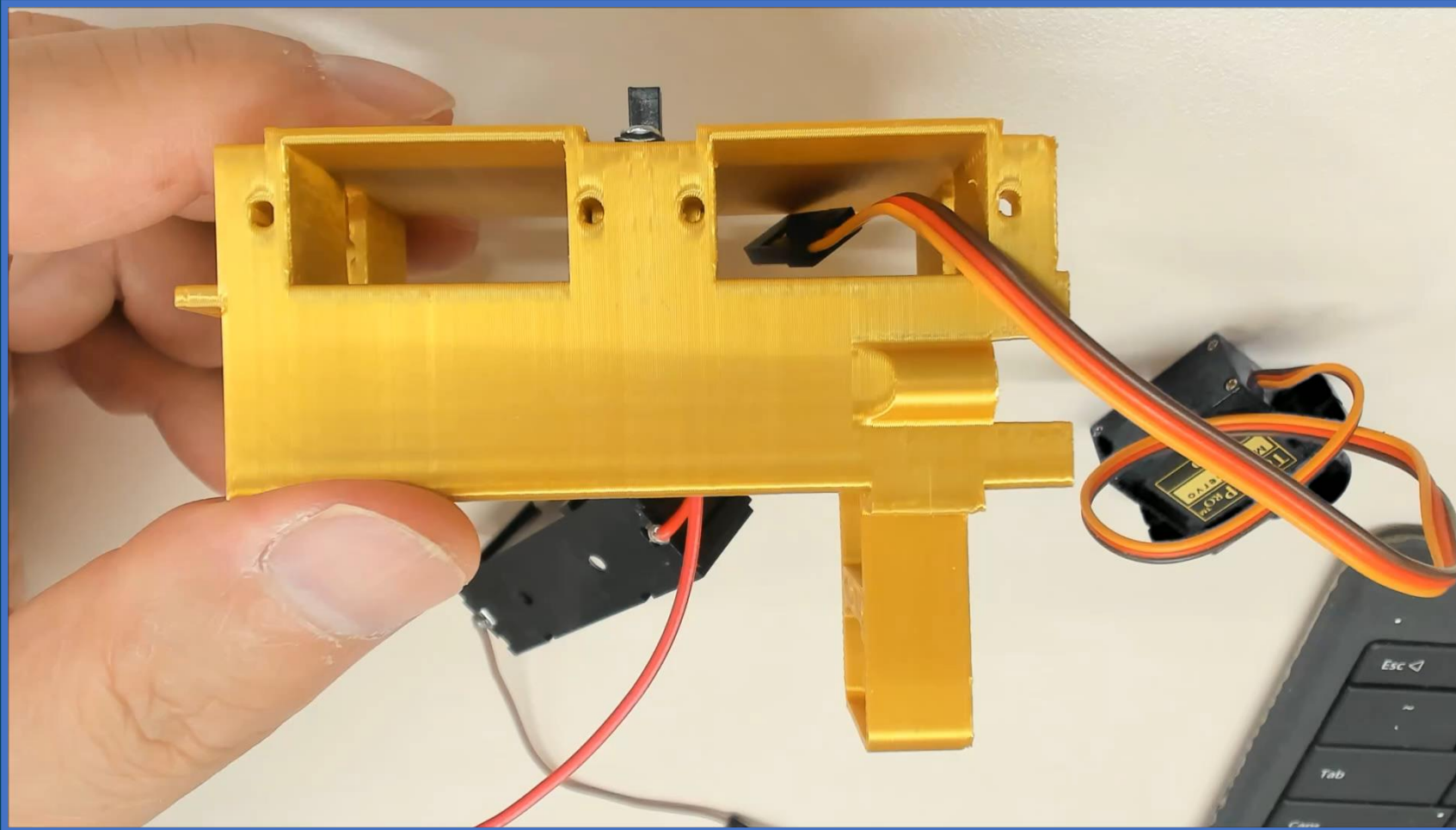
Assembly of the robot:

- 1 Secure the On/off switch



Assembly of the robot:

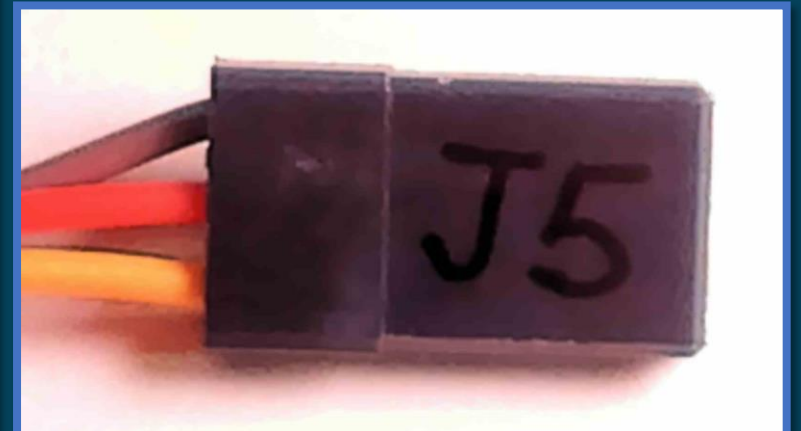
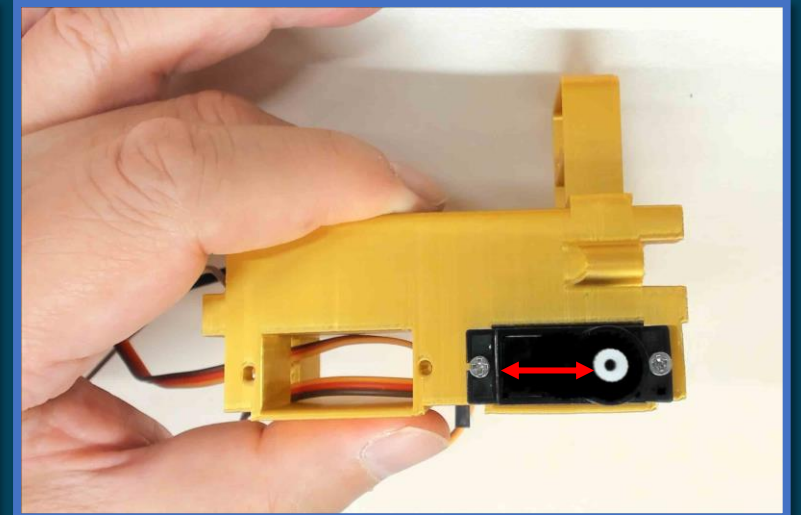
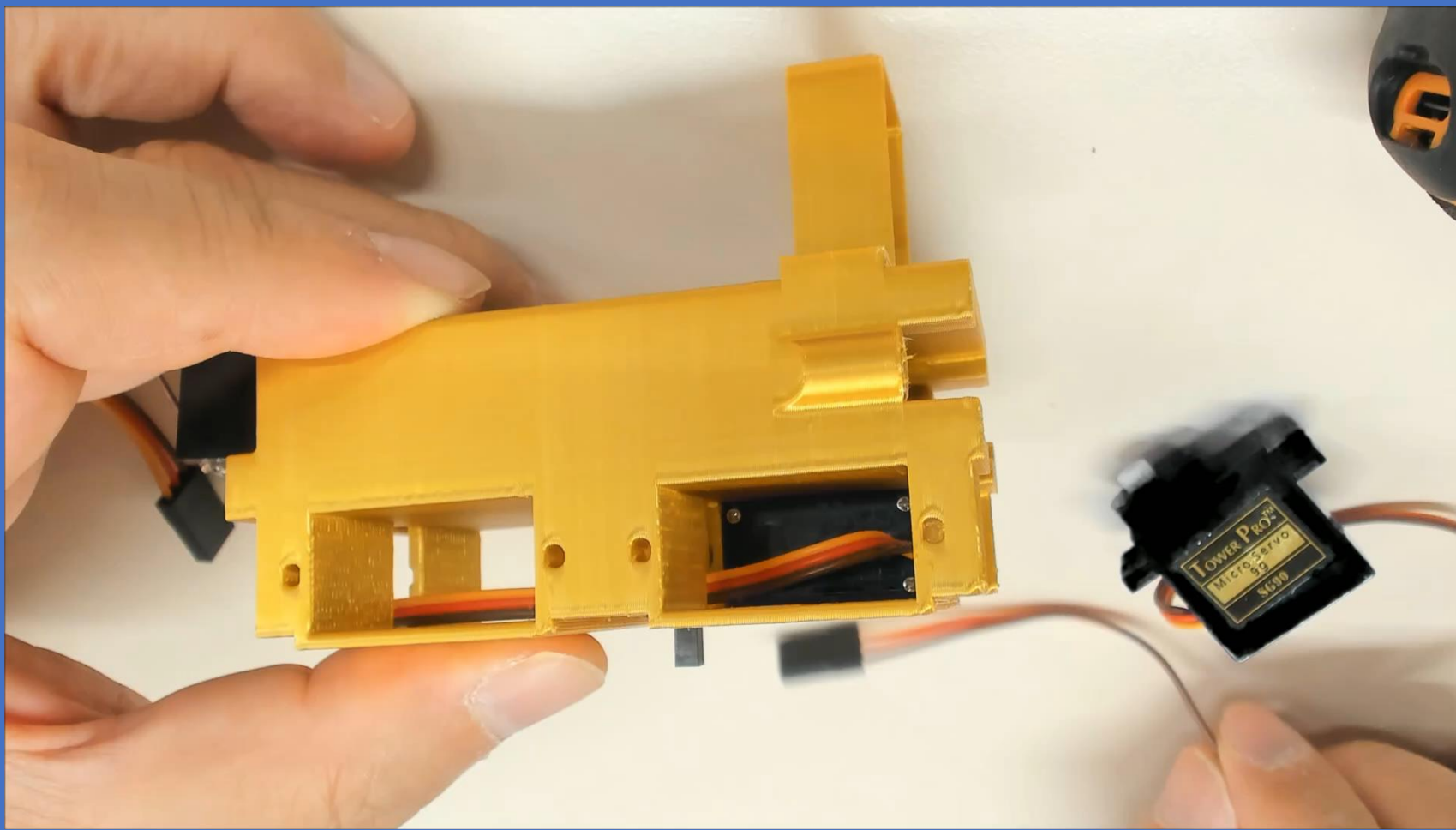
- 2 Place the servo motor of the **left front leg** **J7**



Assembly of the robot:

- ③ Place the servo motor of the **right front** leg

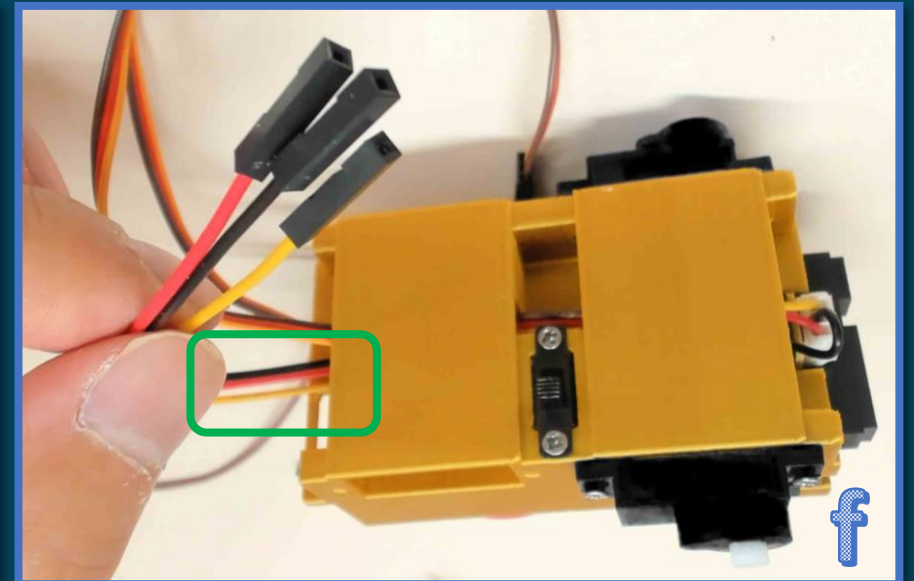
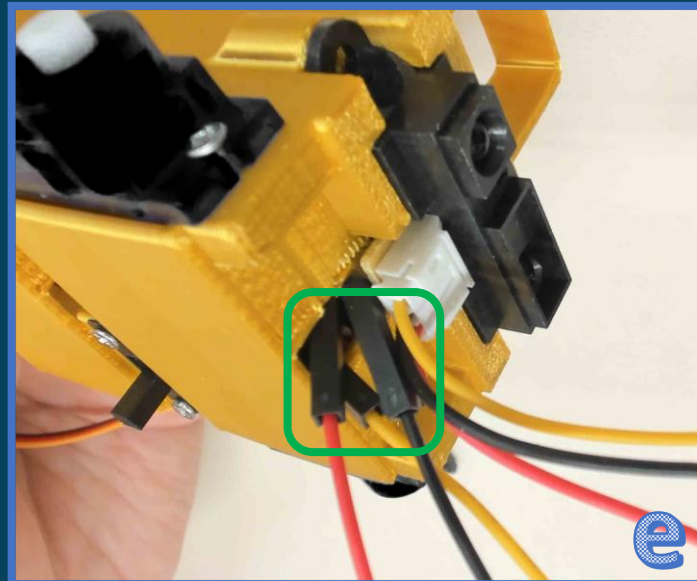
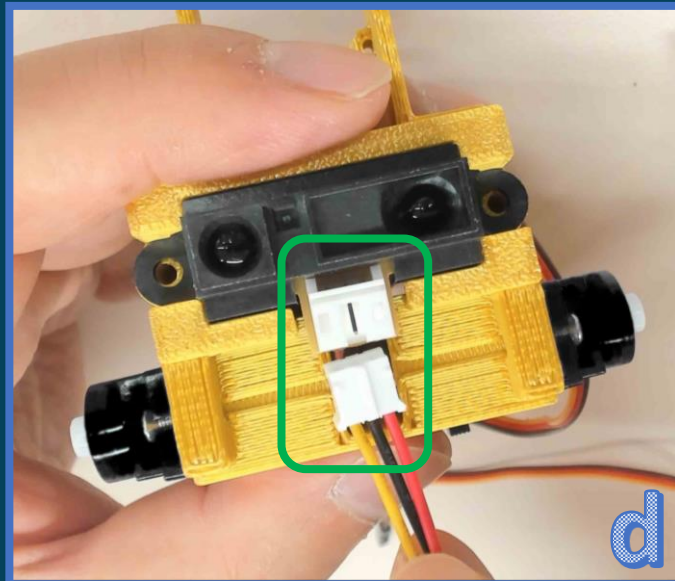
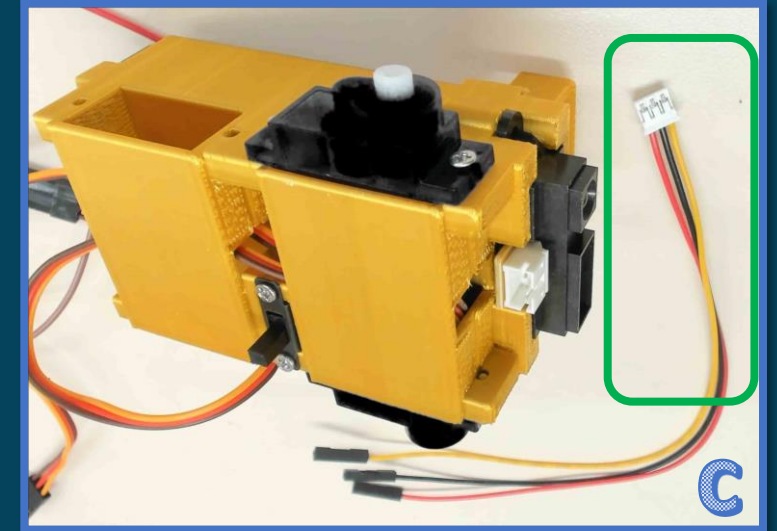
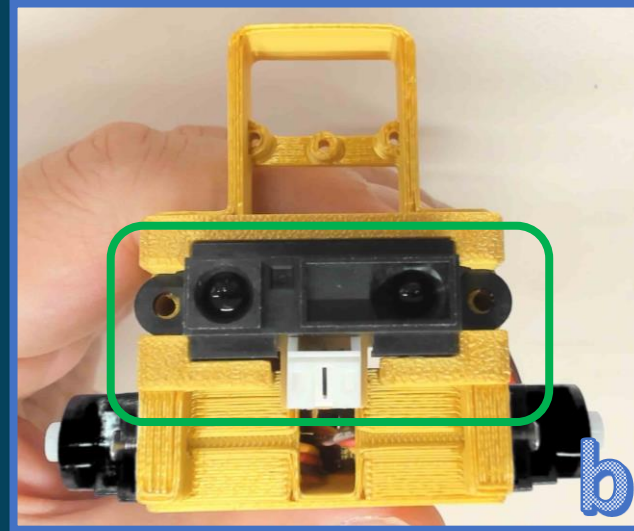
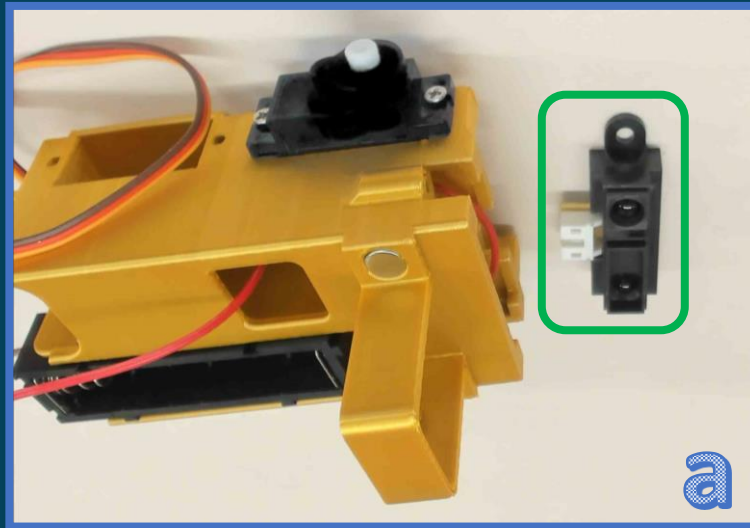
J5



Assembly of the robot:

- 4 Place the **PSD** (Position sensitive detector)

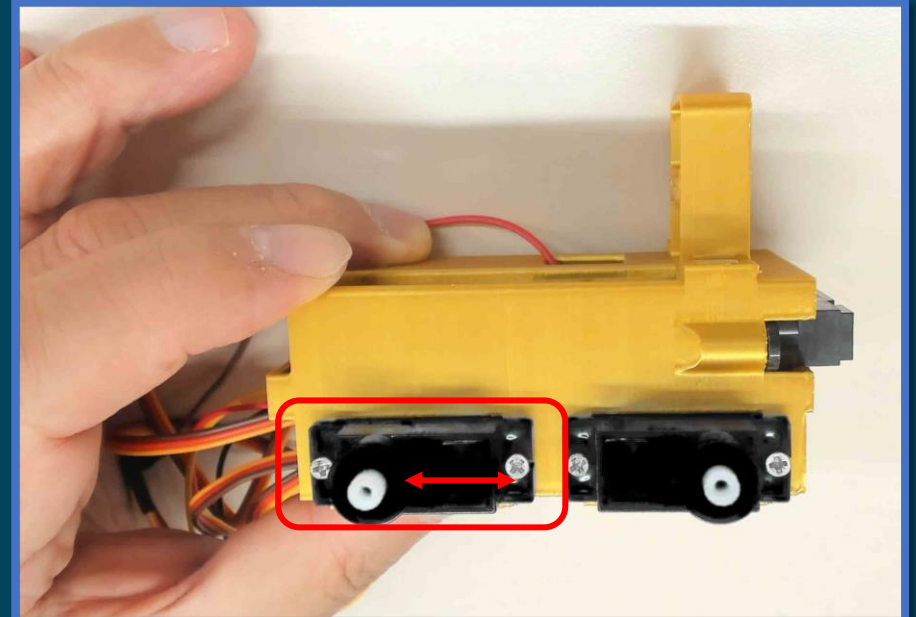
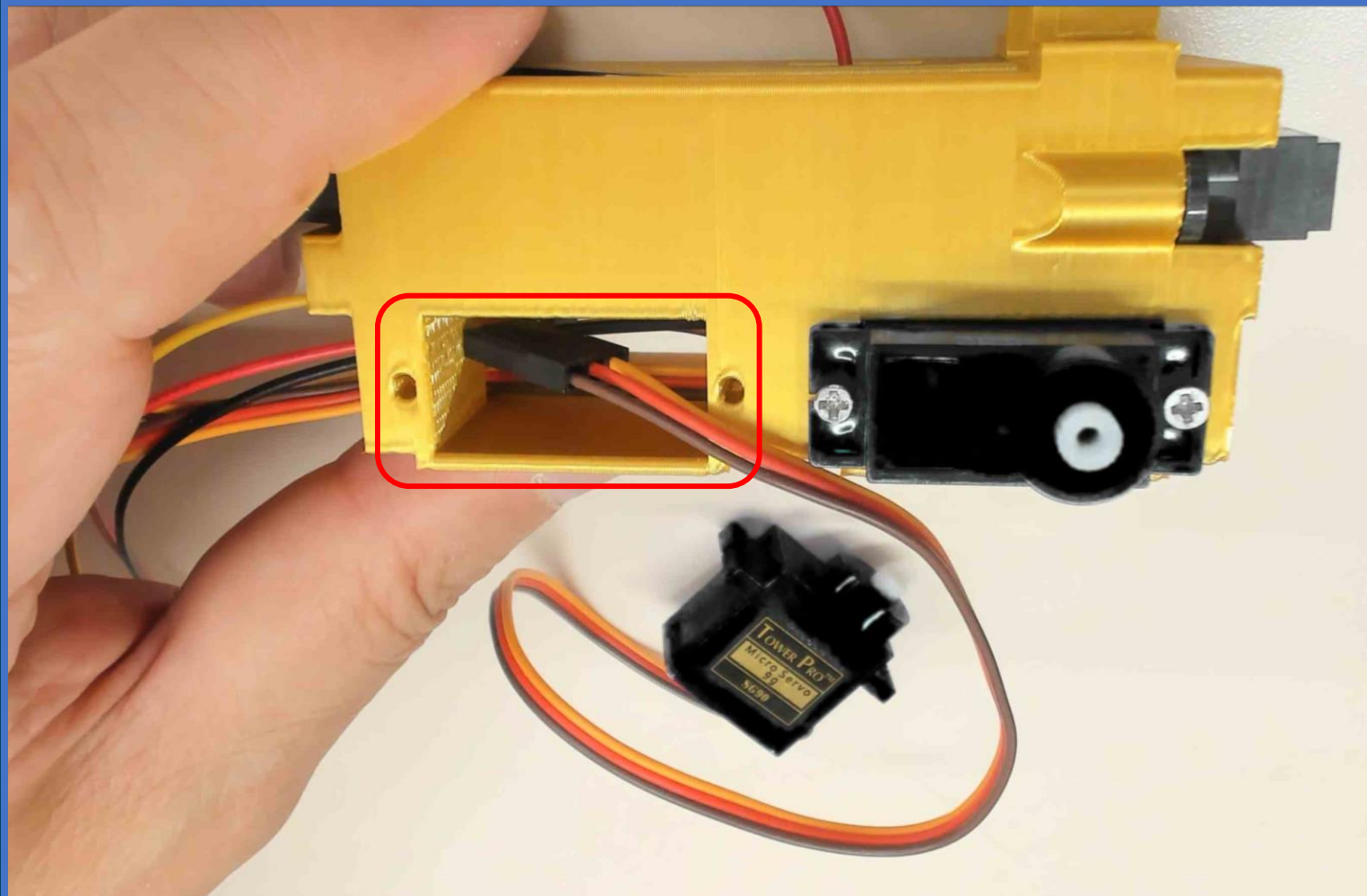
J4



Assembly of the robot:

- 5 Place the servo motor of the **right back** leg

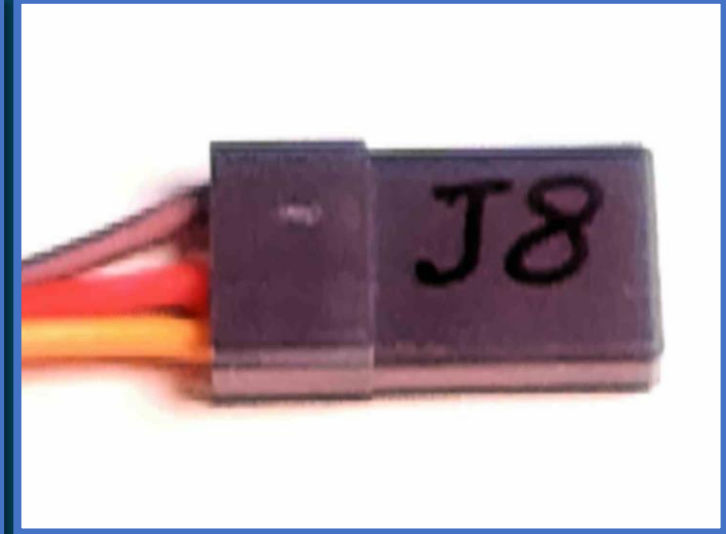
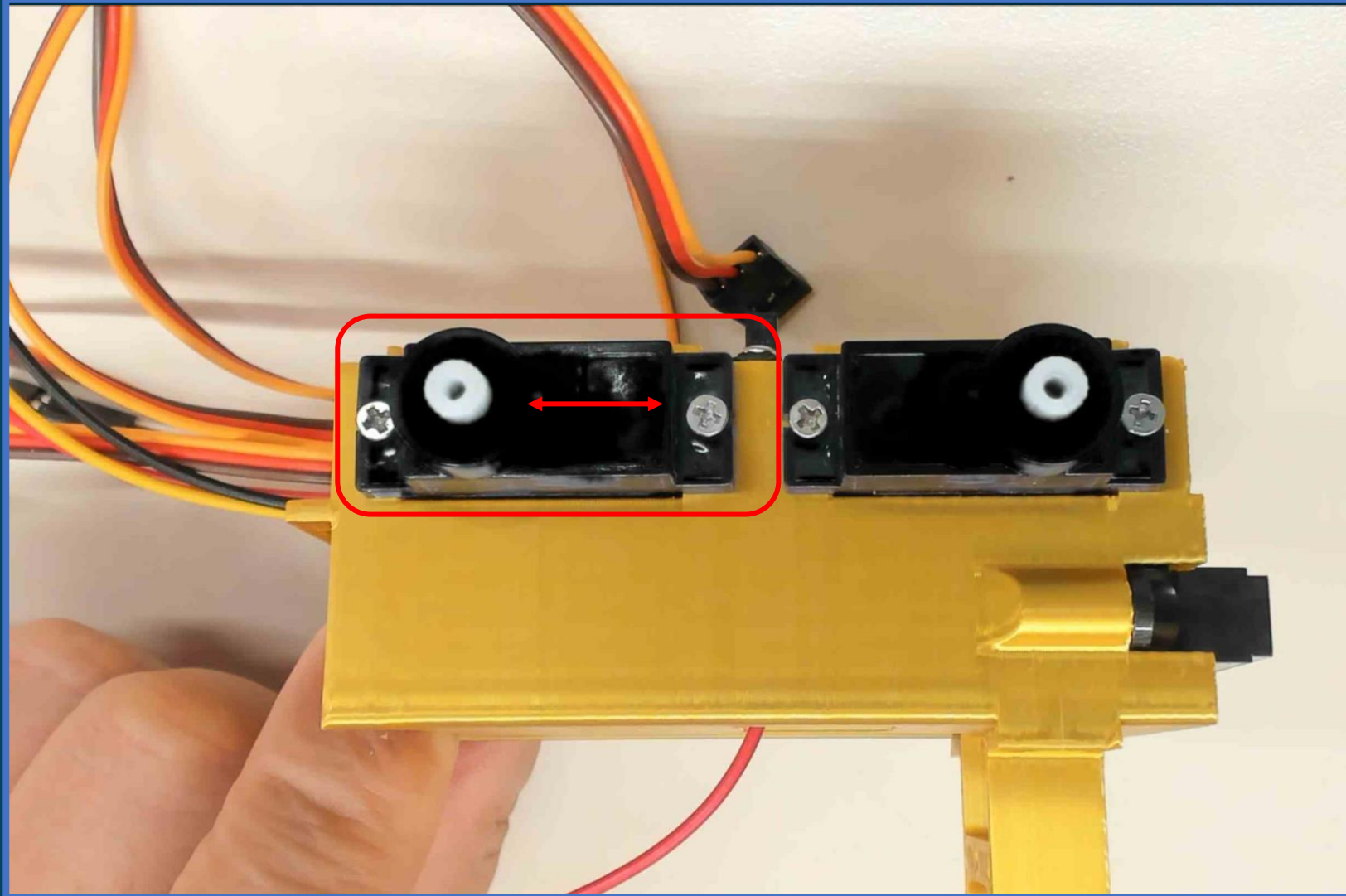
J6



Assembly of the robot:

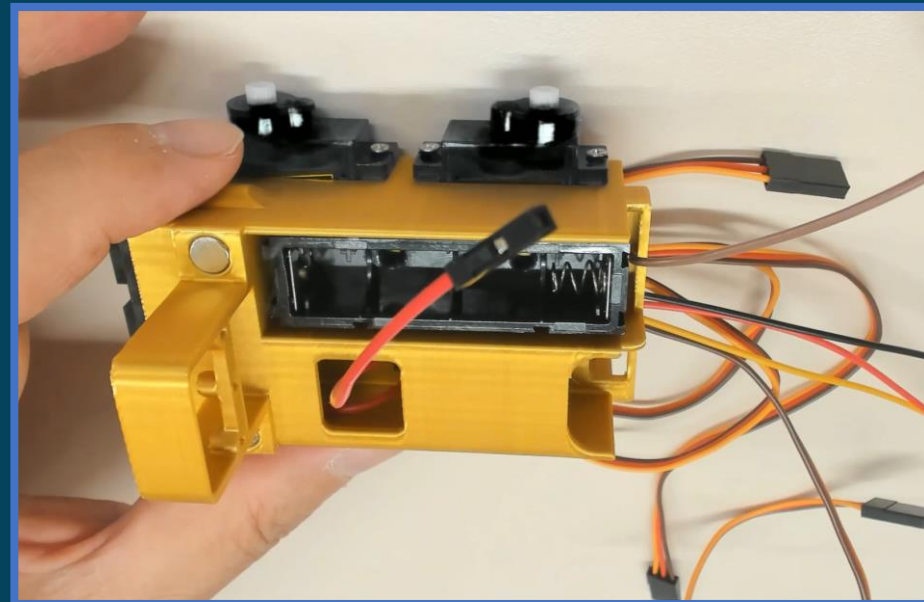
- 6 Place the servo motor of the **left back** leg

J8

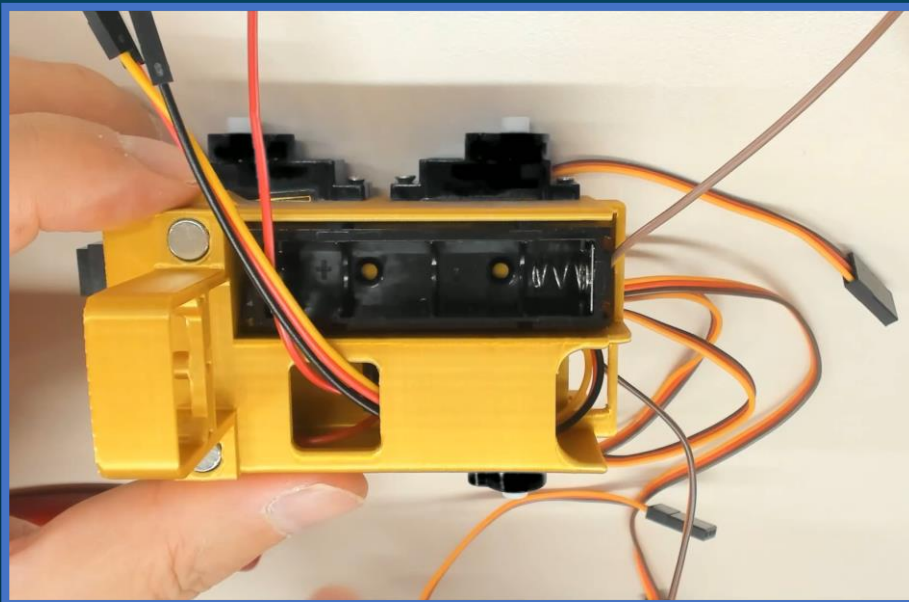


Assembly of the robot:

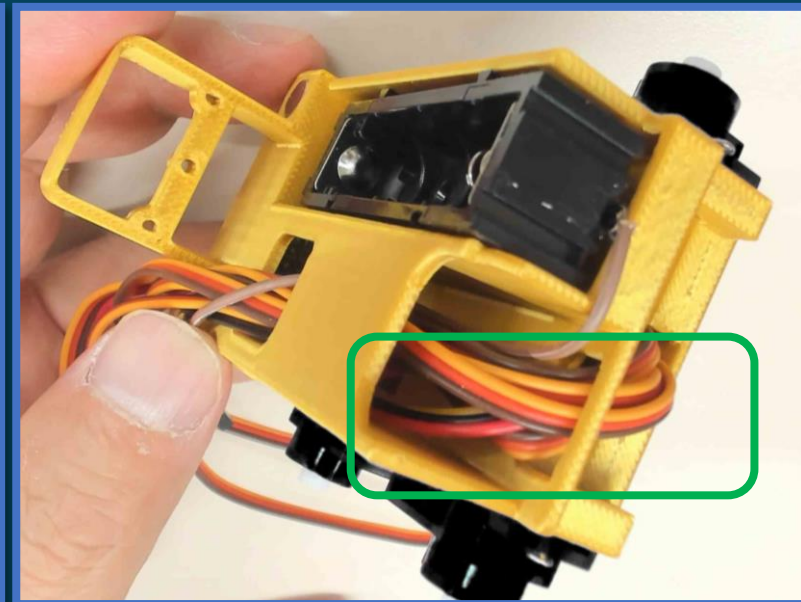
- 7 Store the wires inside the duct



a



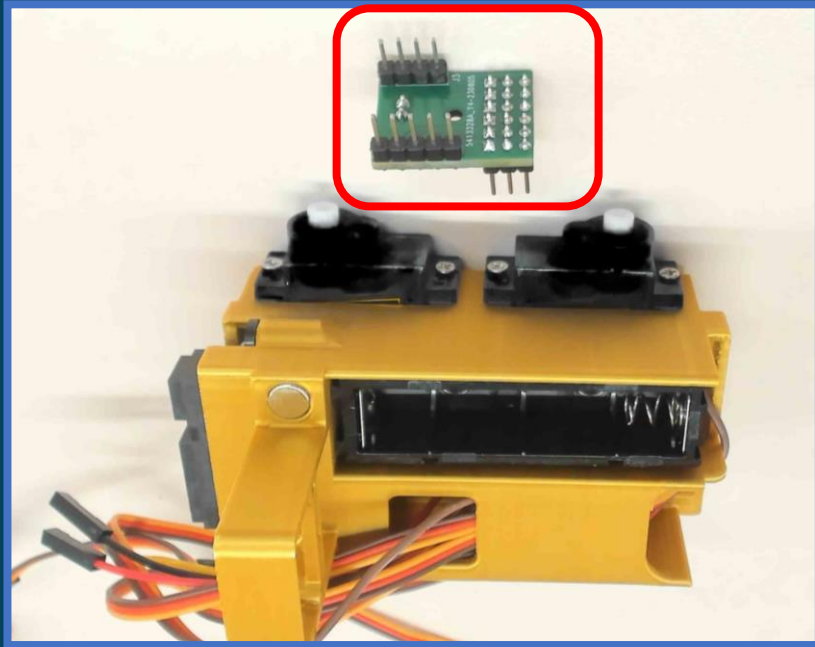
b



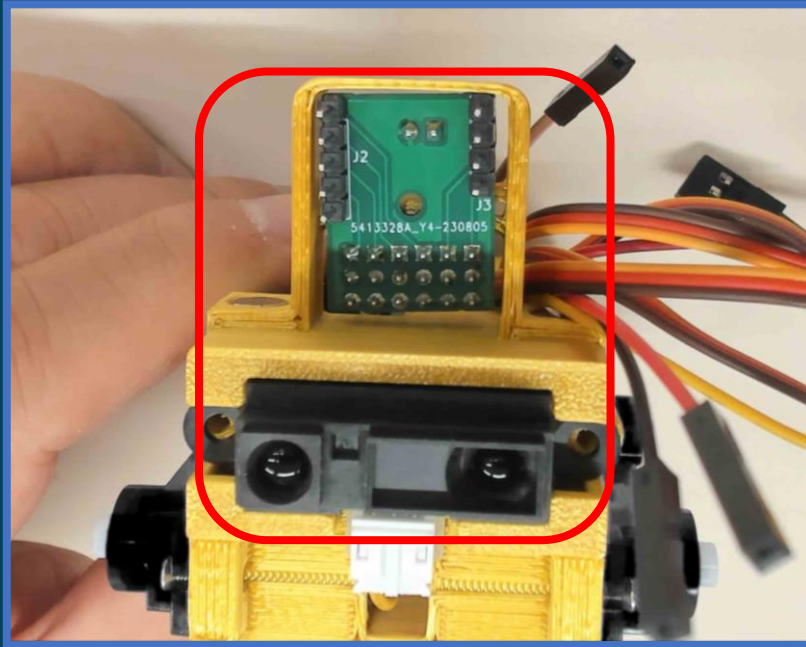
c

Assembly of the robot:

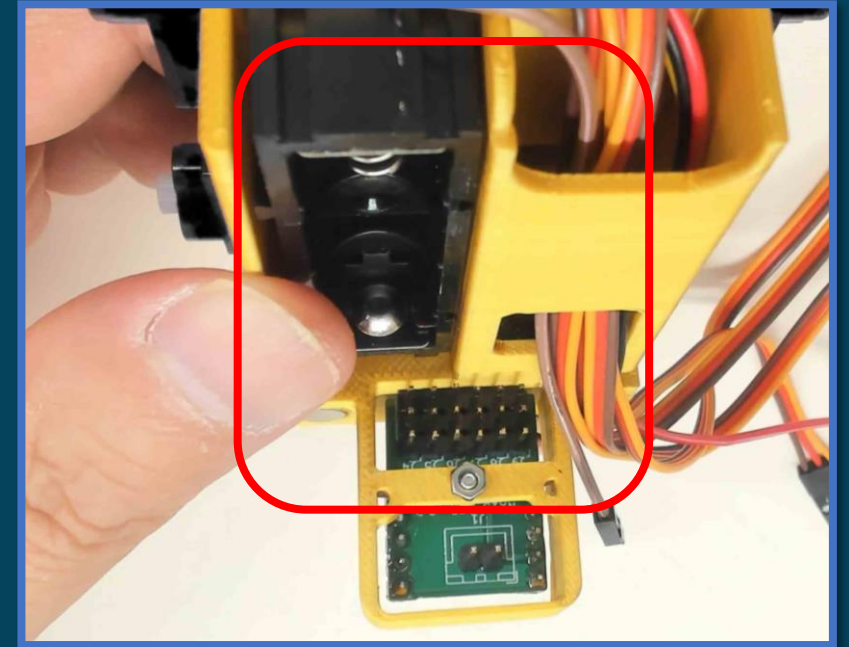
- 8 Place the PCB (Printed circuit board)



a



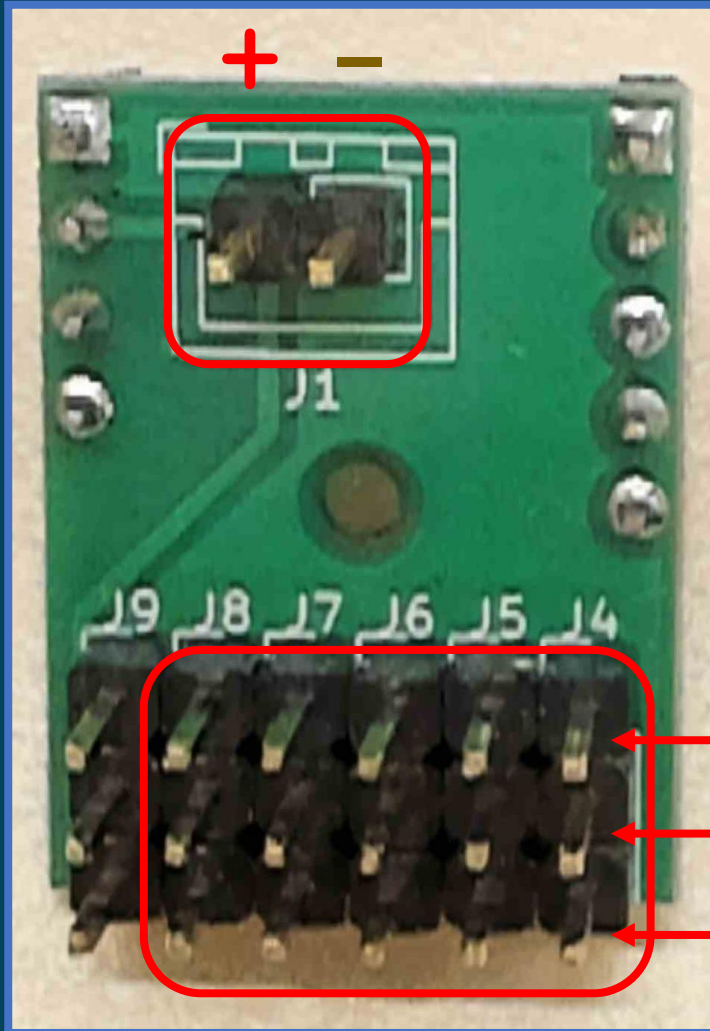
b



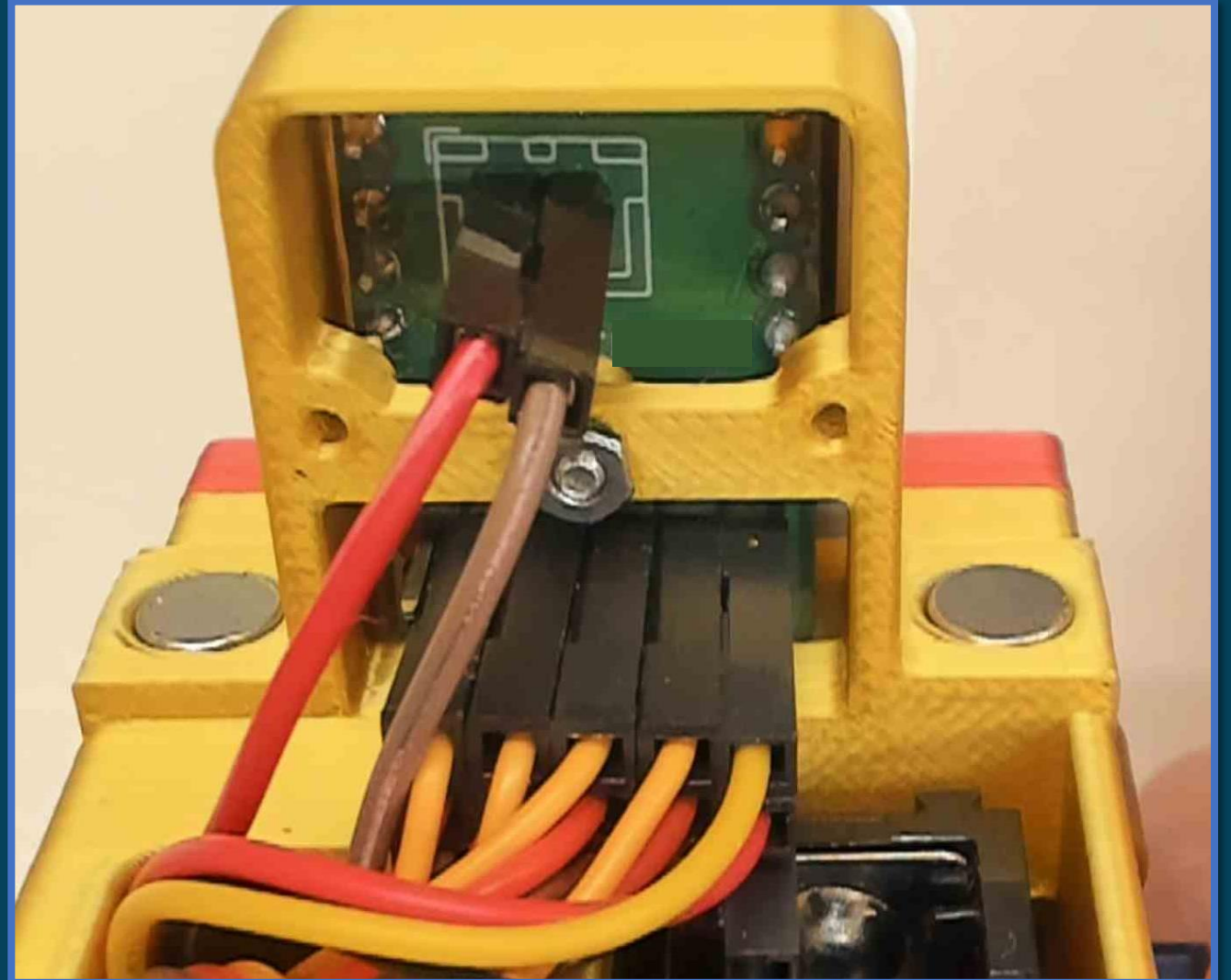
c

Assembly of the robot:

- 9 Attach the jumper wires to the connector pins

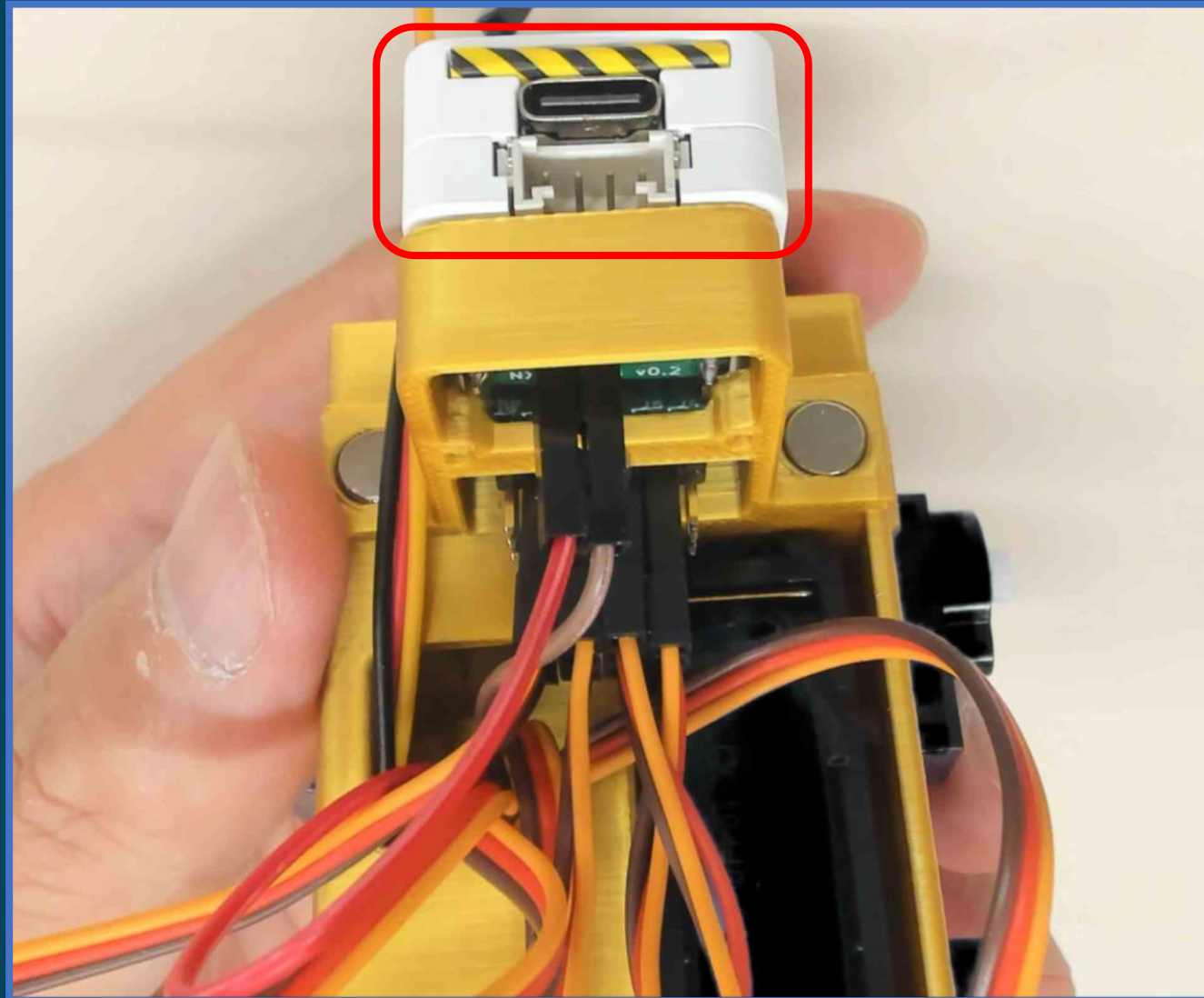


Orange
Red
Brown



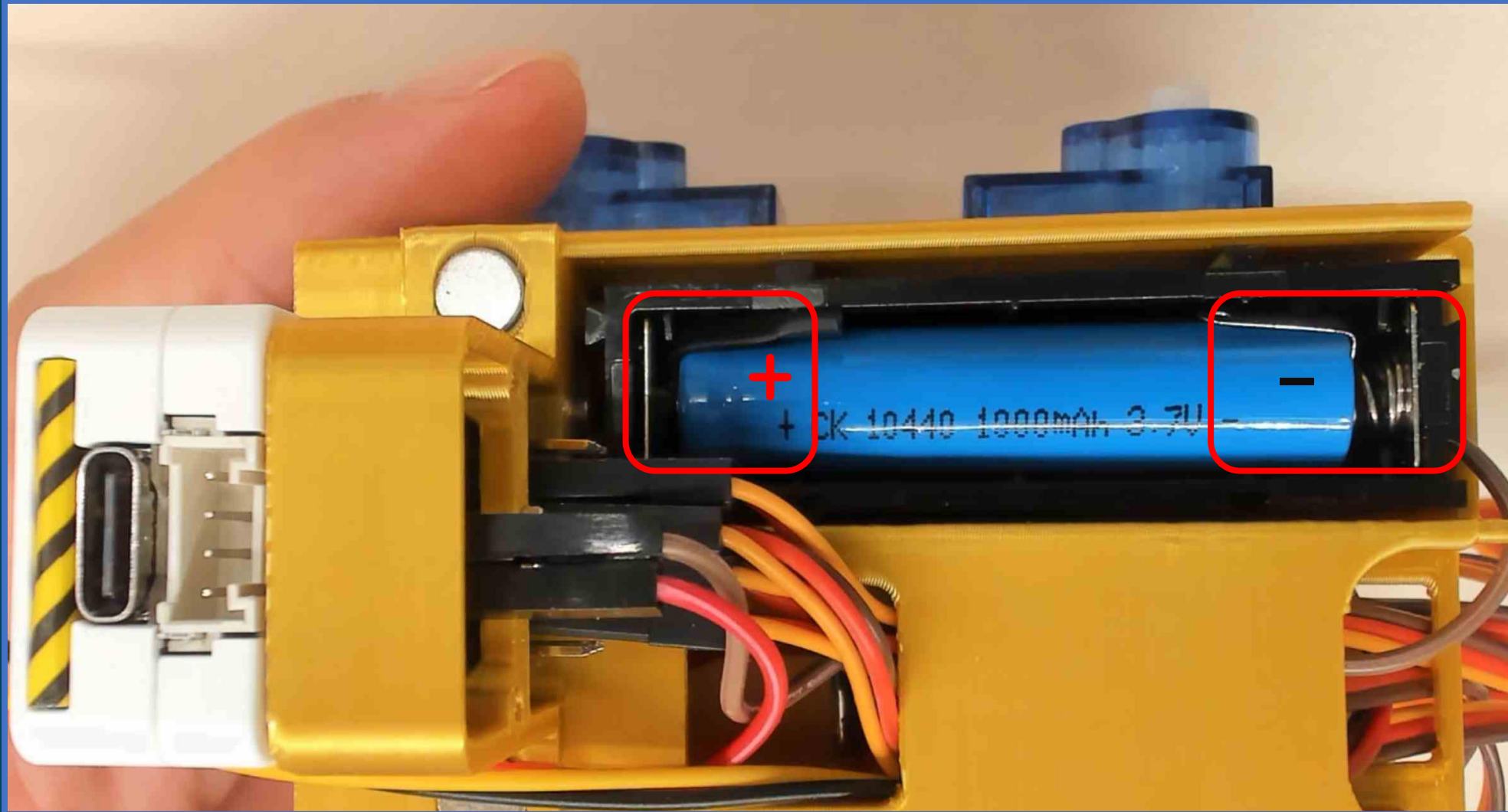
Assembly of the robot:

- 10 Attach the MCU (Atom S3) to the connector pins



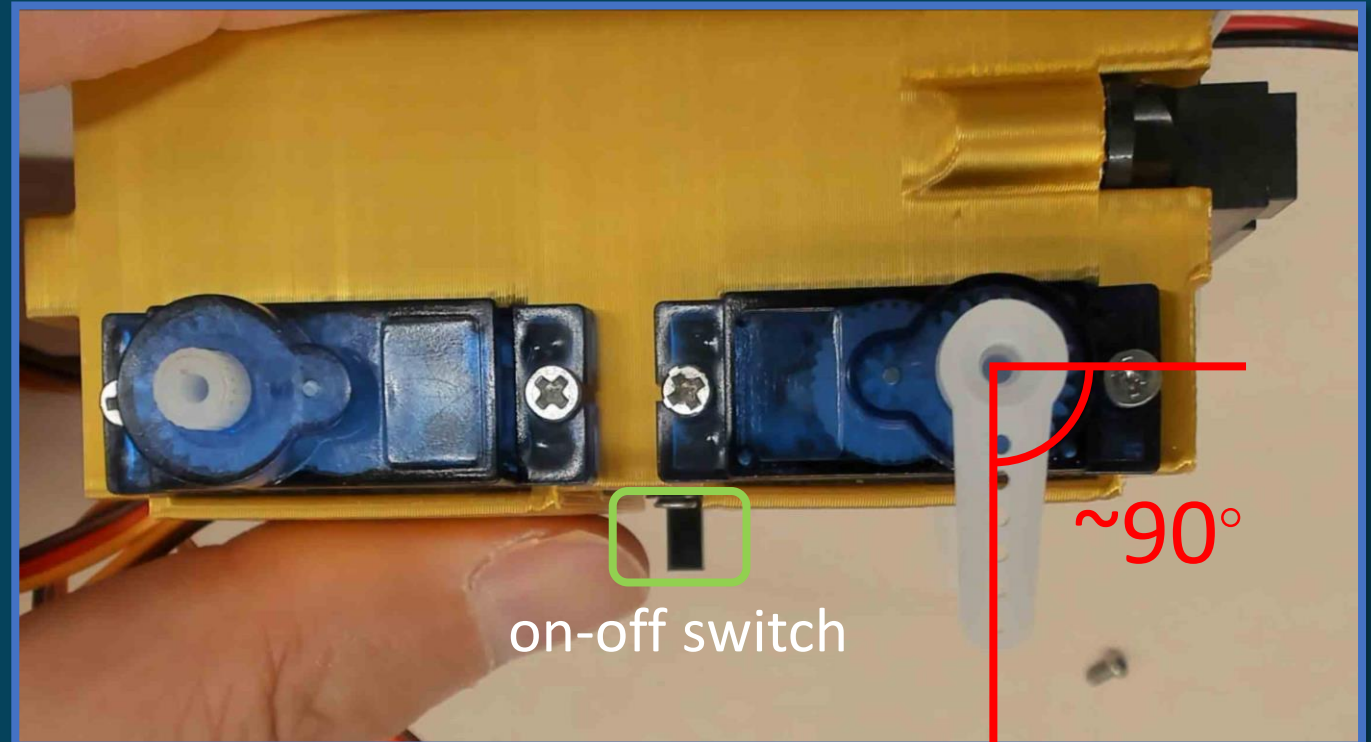
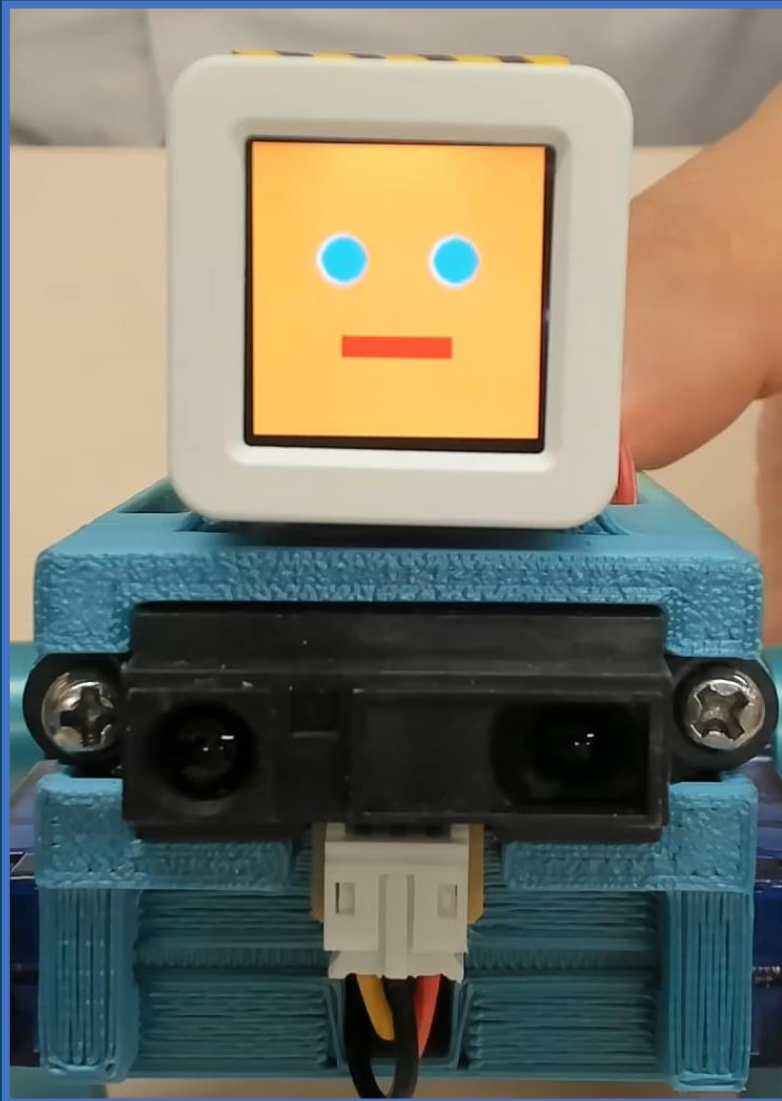
Assembly of the robot:

- 11 Place the battery to the battery case



Assembly of the robot:

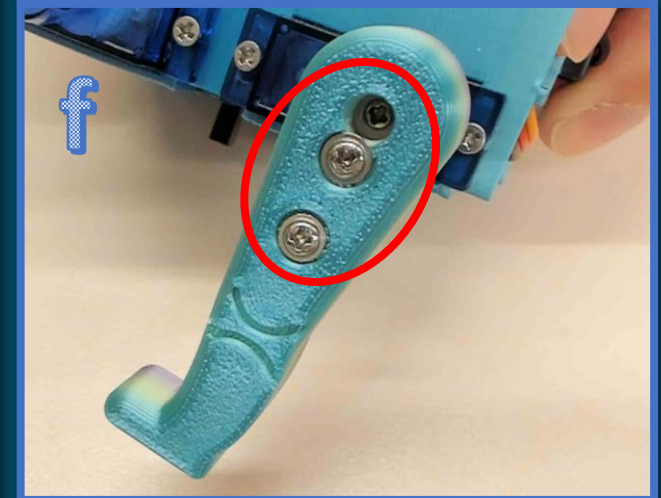
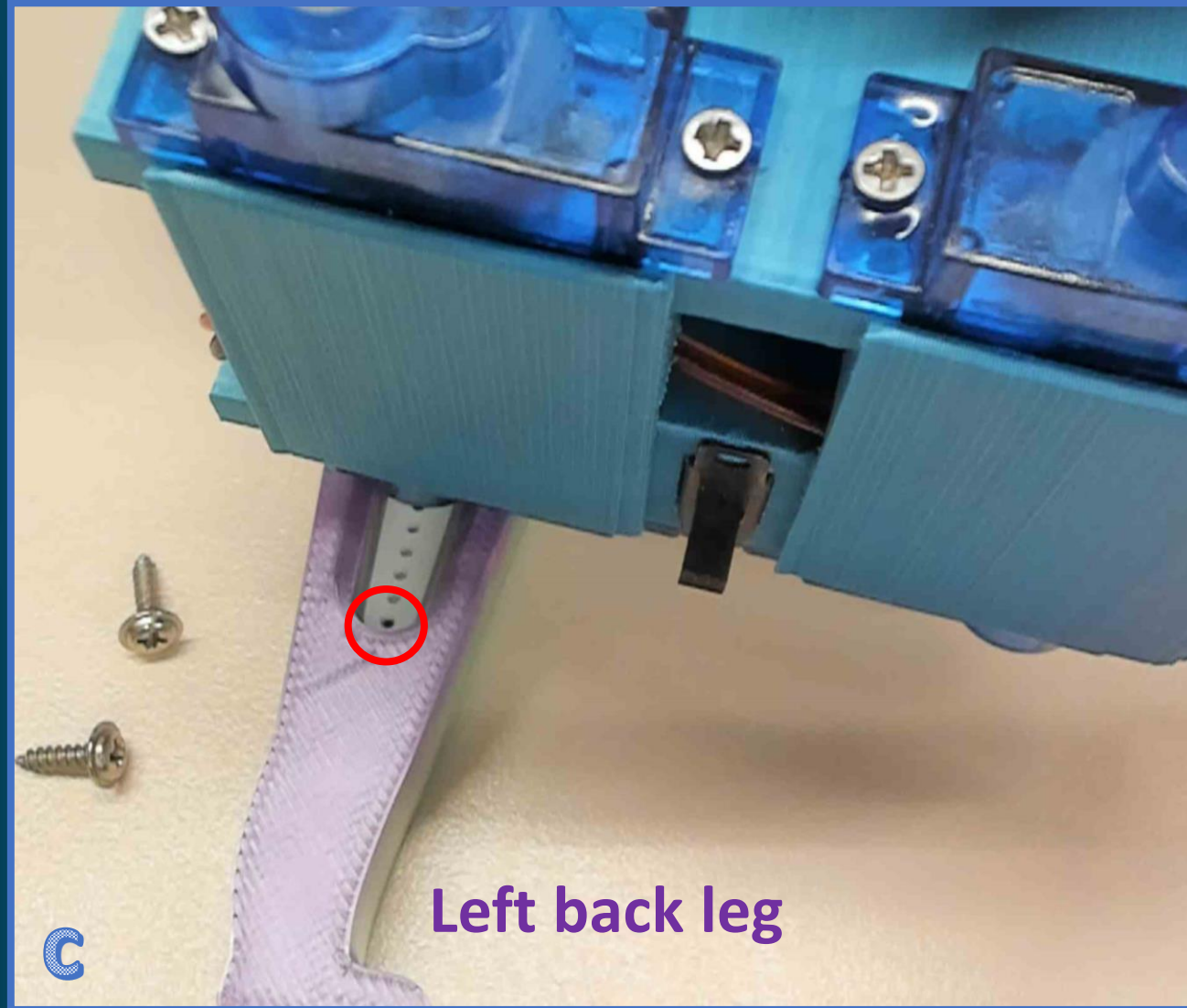
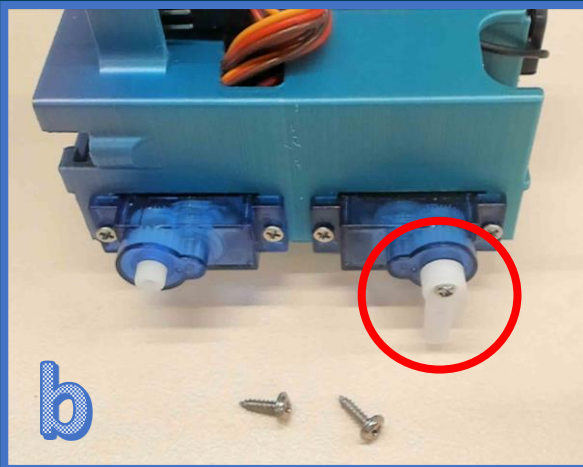
- 12 Switch on the robot to reset the home position of all servo motor shafts



Home position

Assembly of the robot:

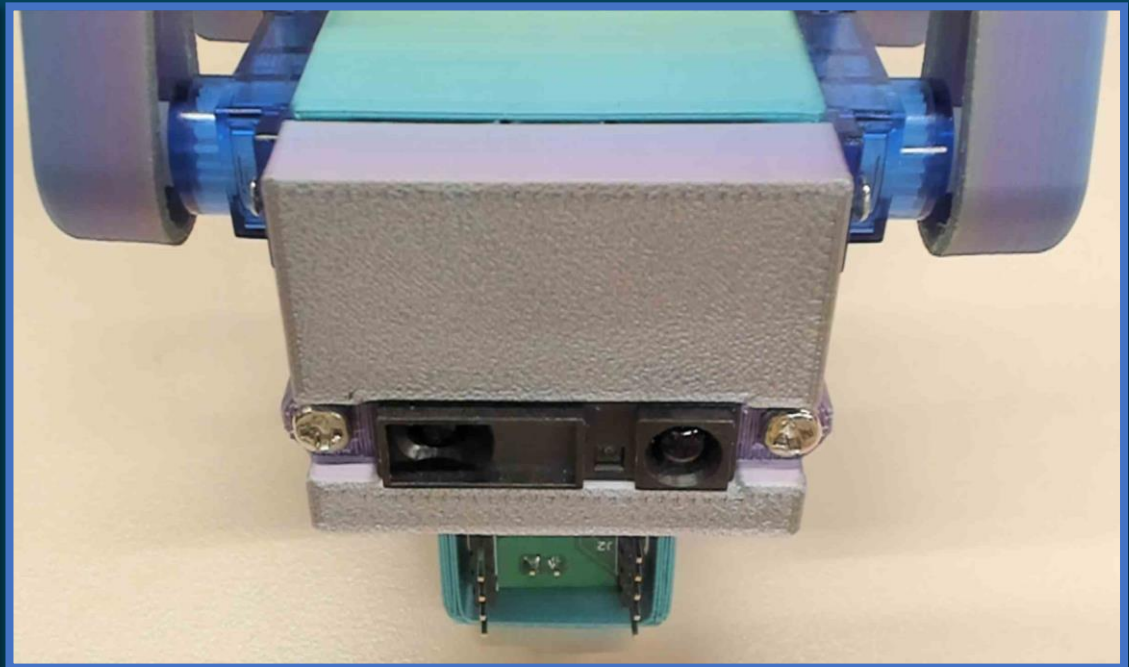
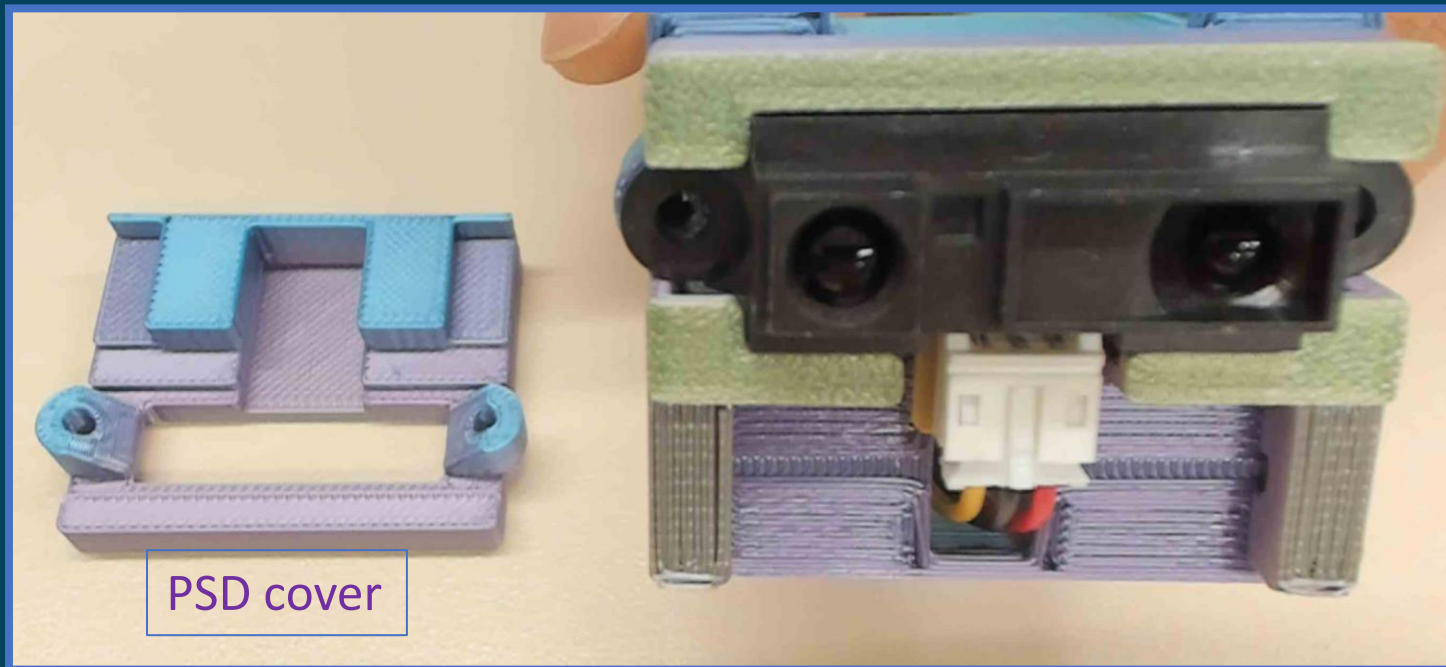
13 Fasten the four legs with screws, one at a time



Assembly of the robot:

14

Fasten the PSD cover with screws

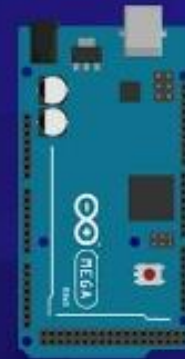


Install the "Dabble" smartphone app for controlling the robot

Dabble: One App for Sensing & Control

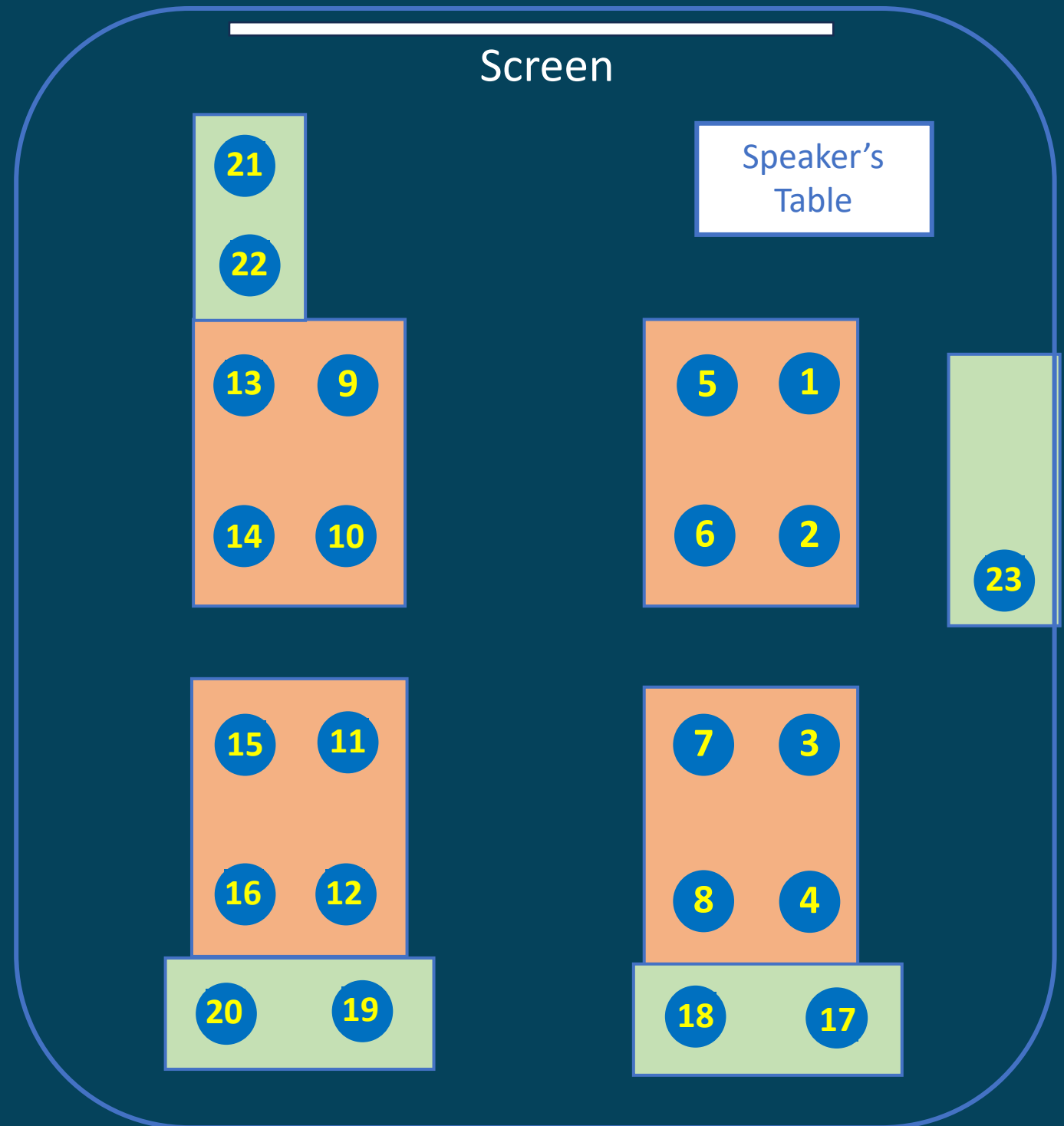
Simplified project-making, wireless hardware control, sensing, IoT & much more.

One App.
Infinite Control.

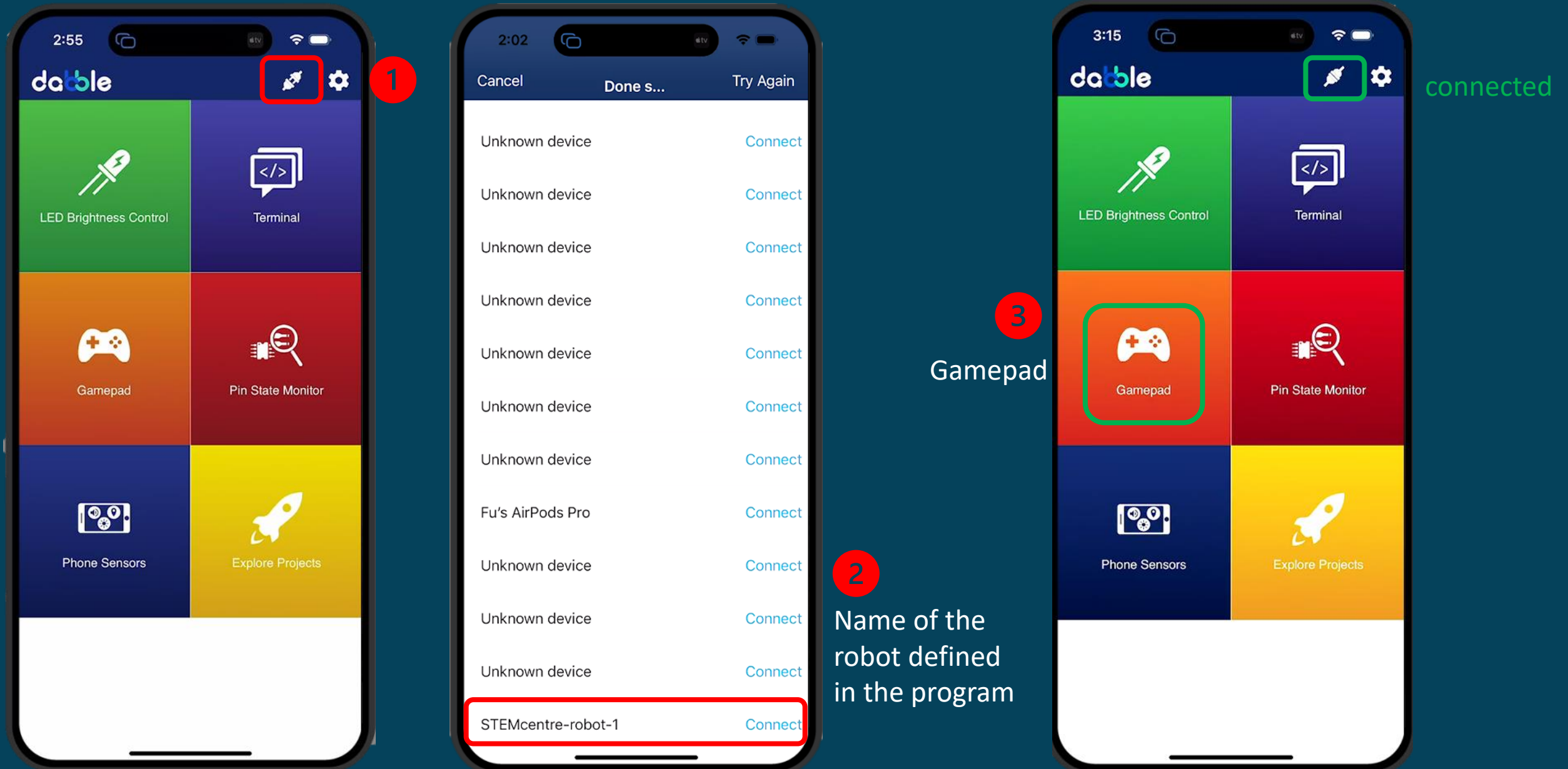


Name of the robot
defined in the program:

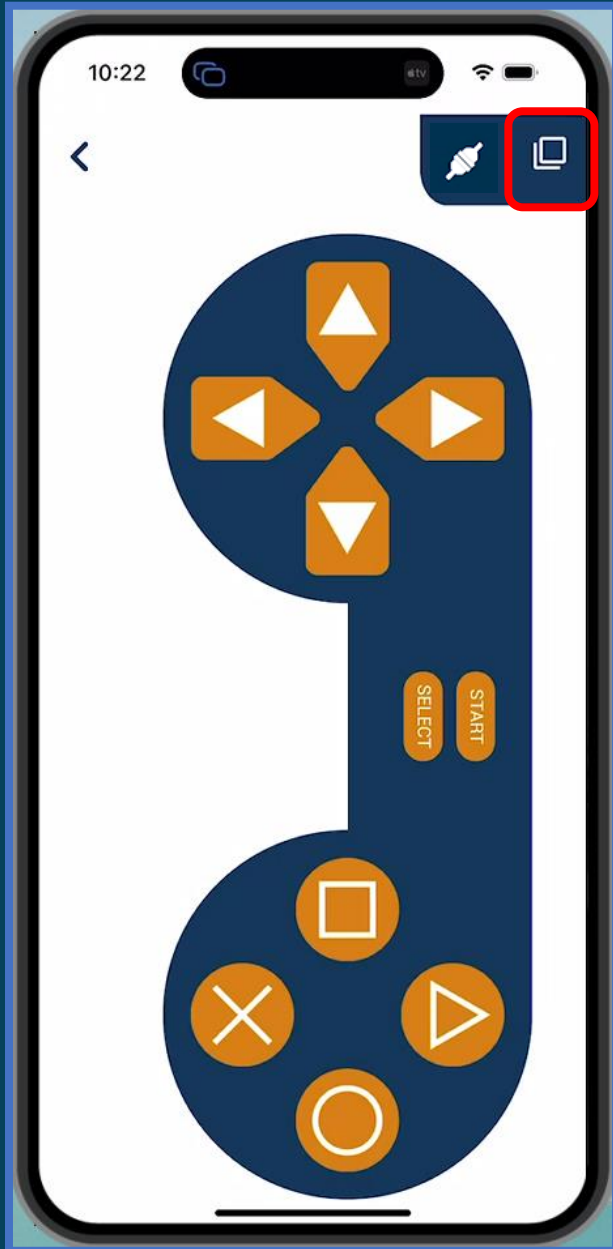
Teacher-robot-**x**



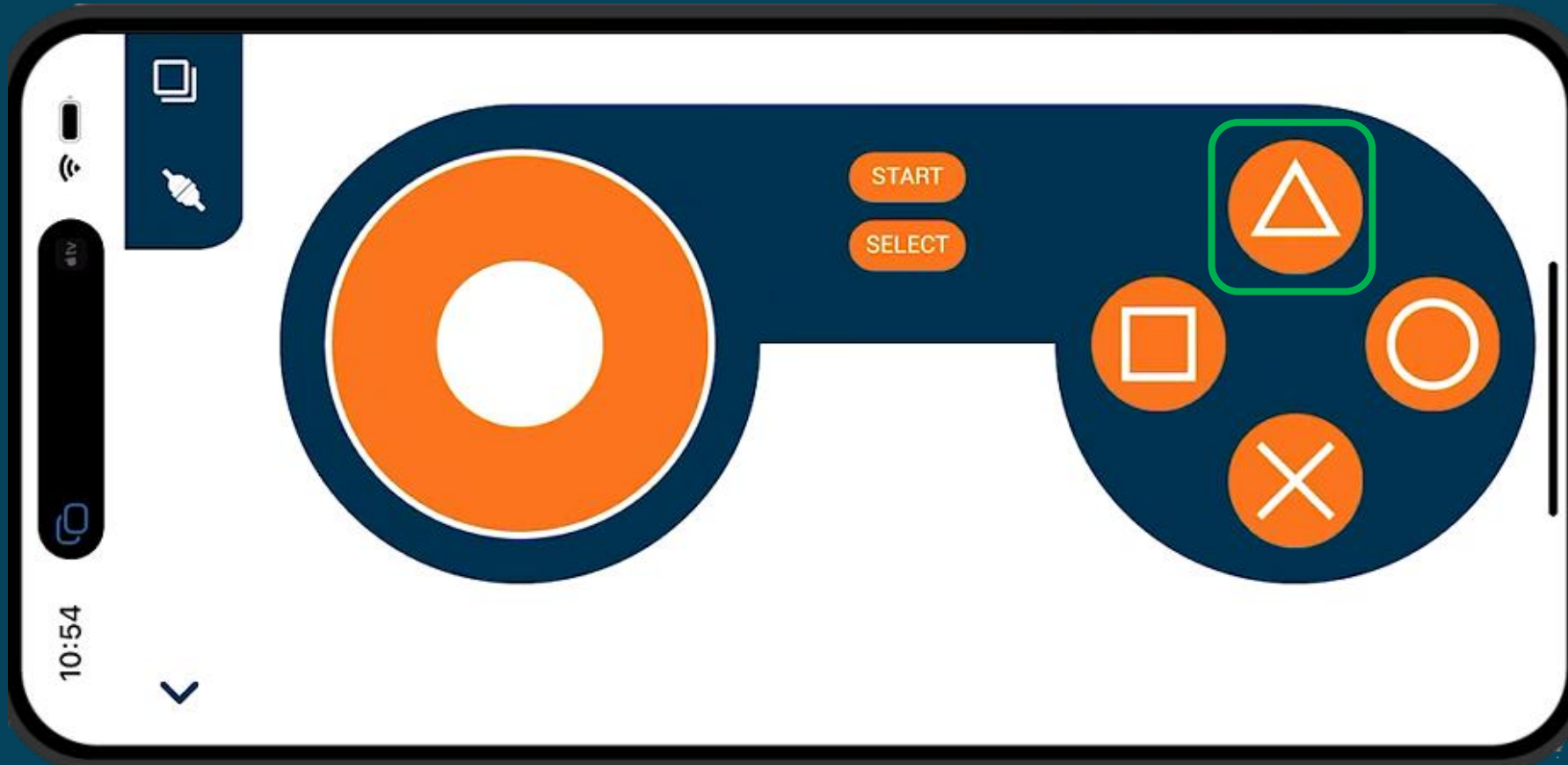
Establish a connection between Dabble and the robot



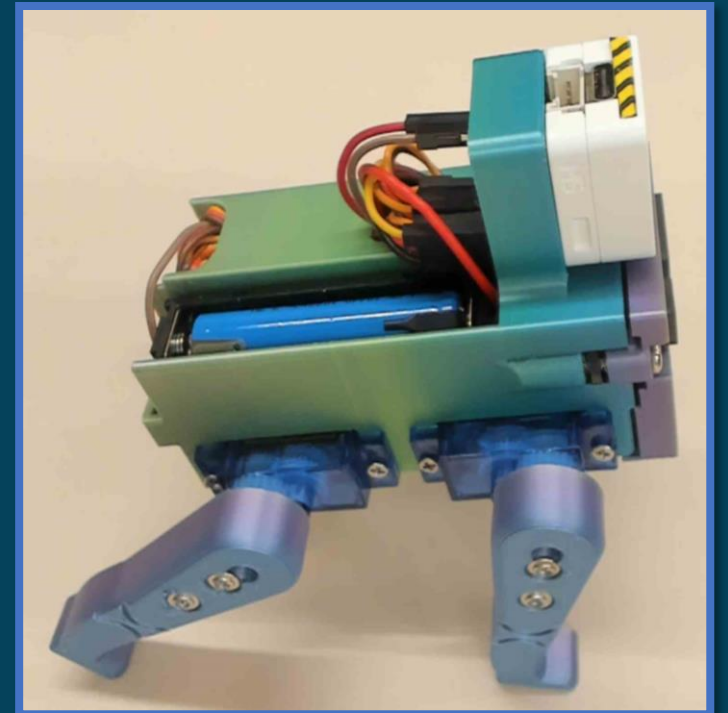
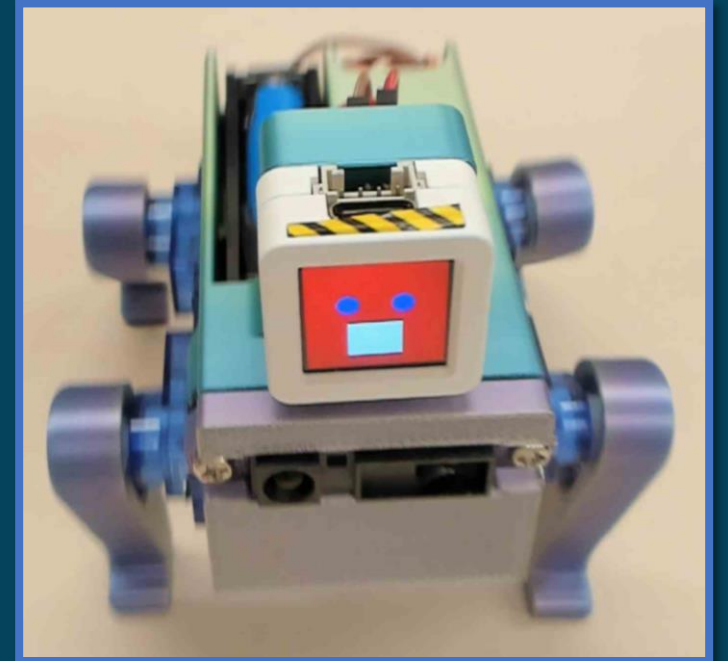
Establish a connection between Dabble and the robot



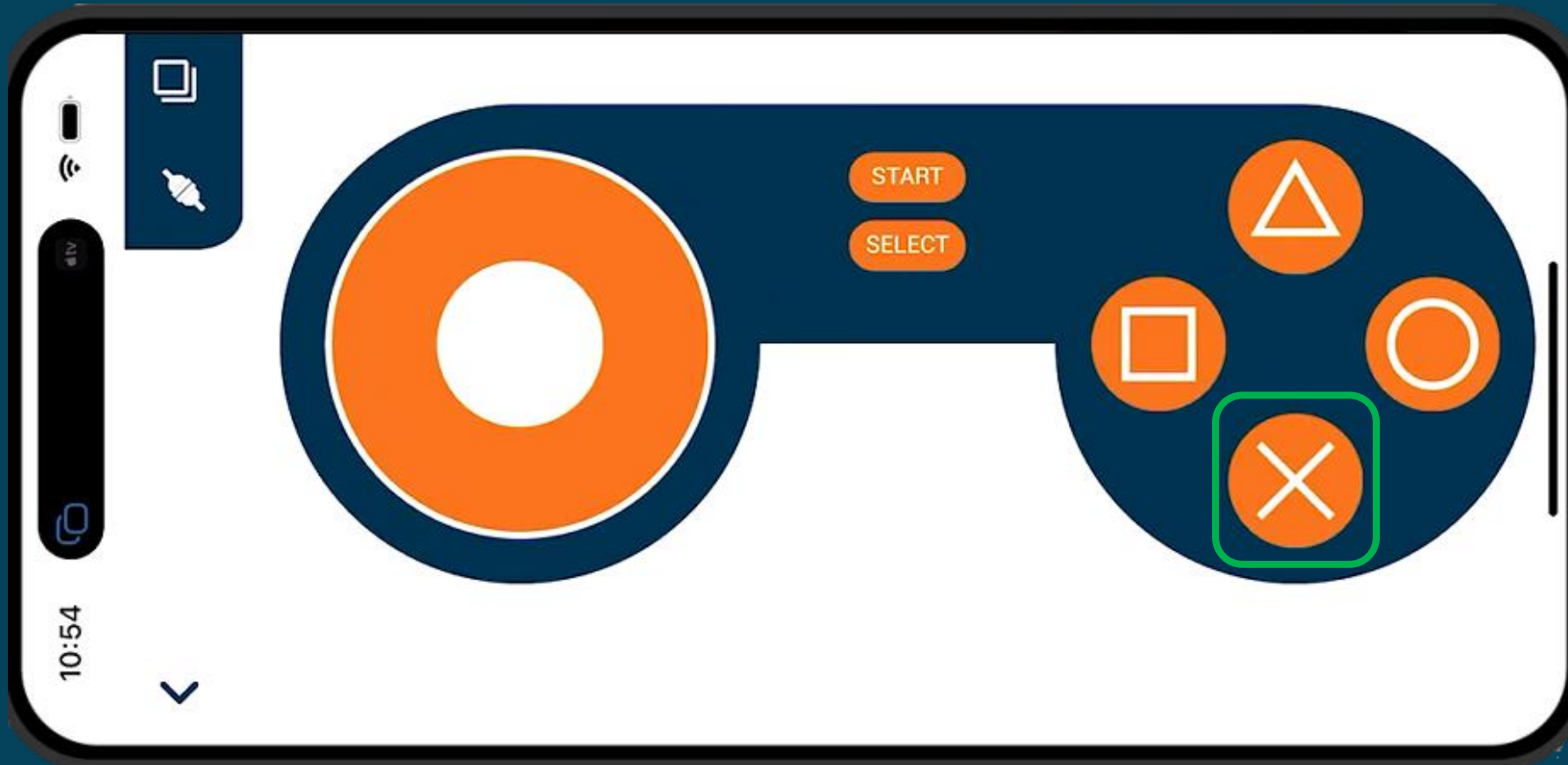
Use Dabble to operate the robot



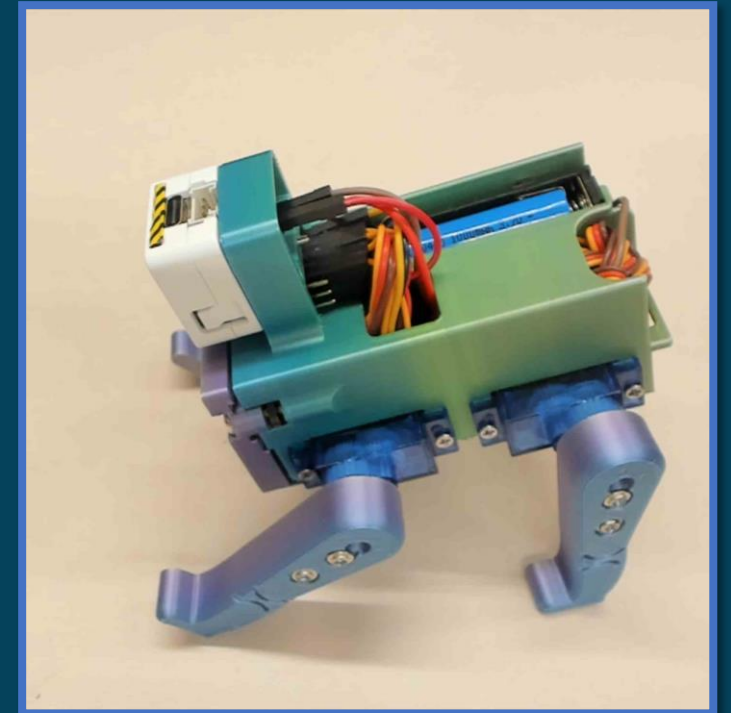
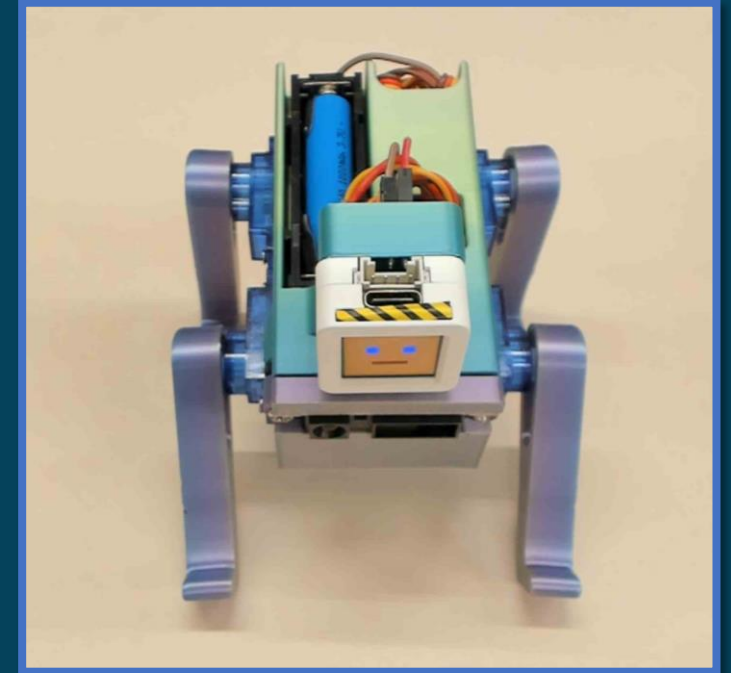
Tilt up



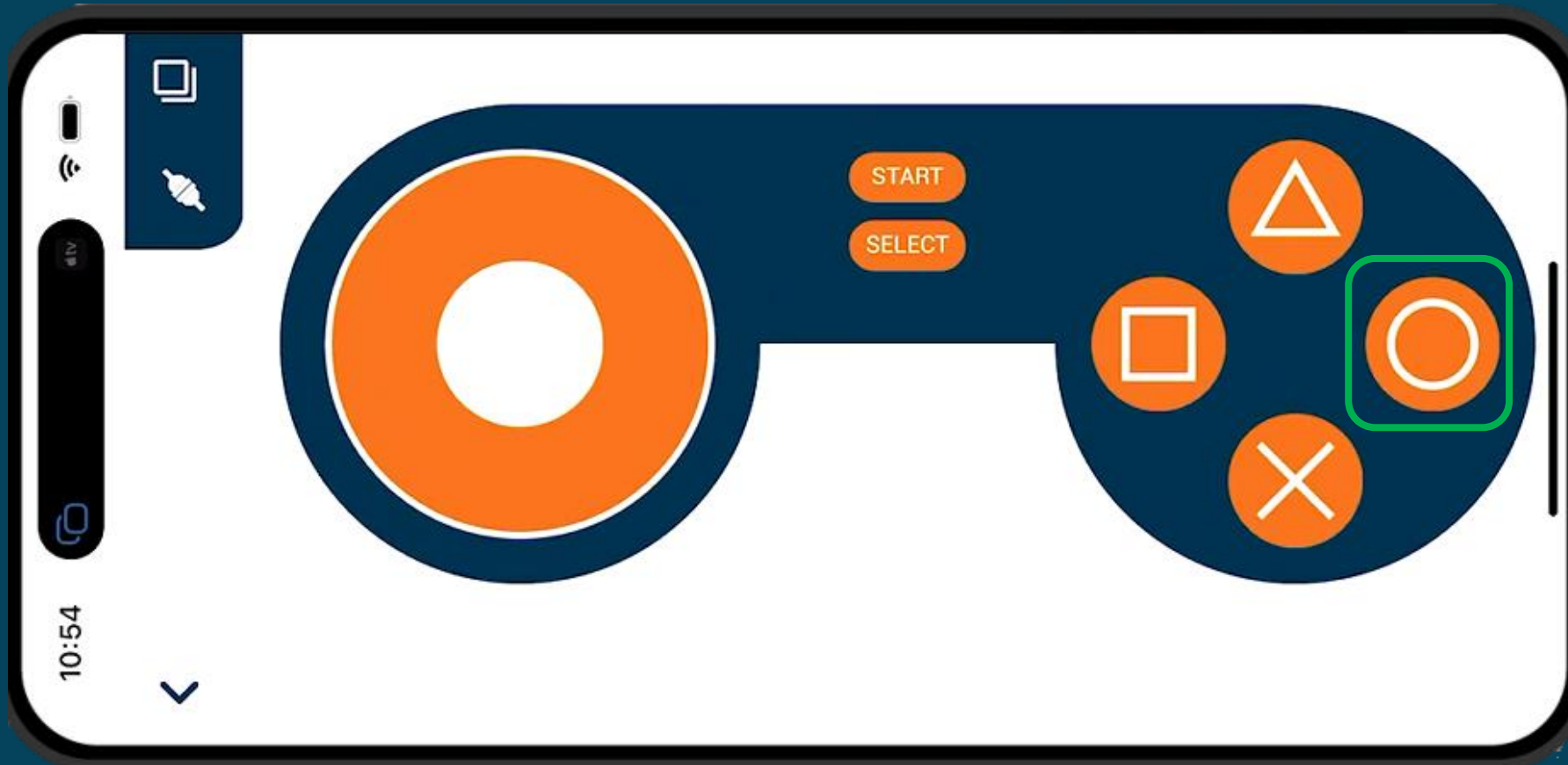
Use Dabble to operate the robot



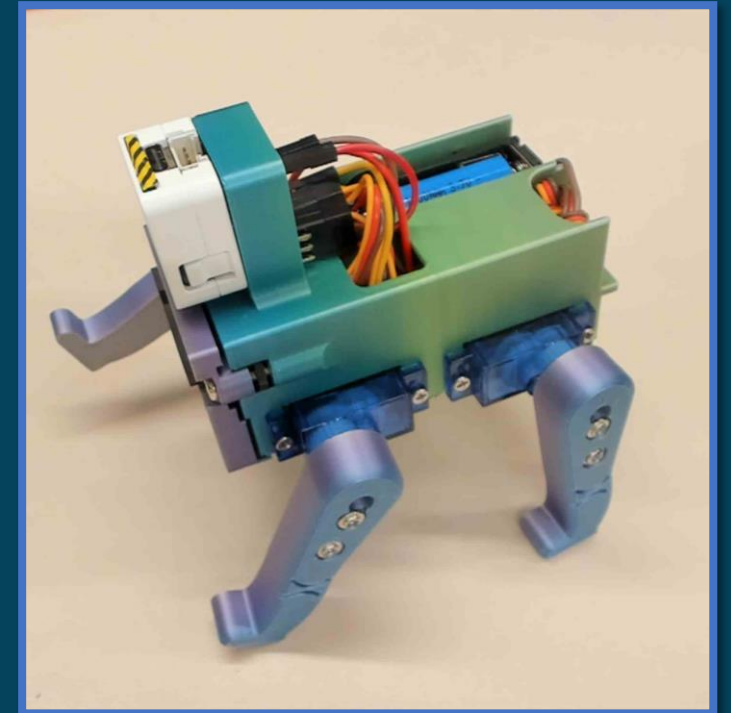
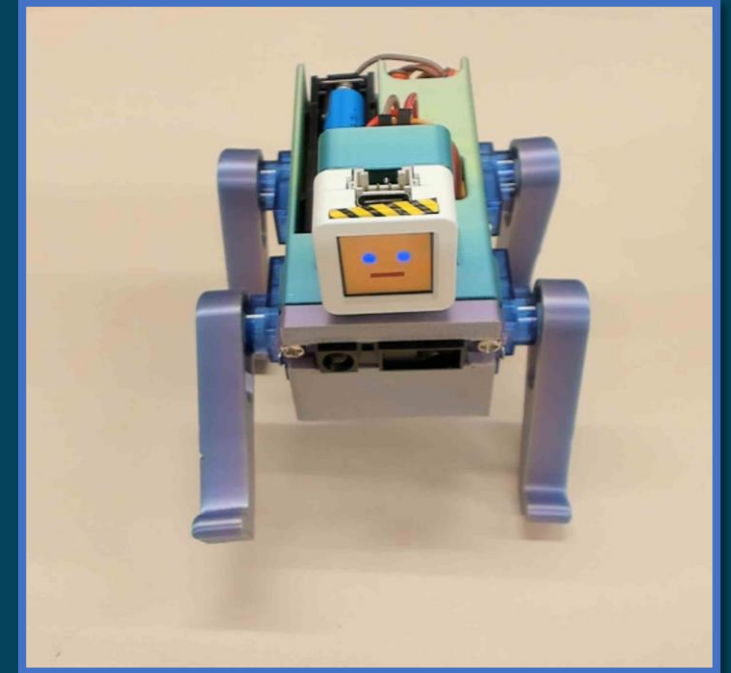
Tilt down



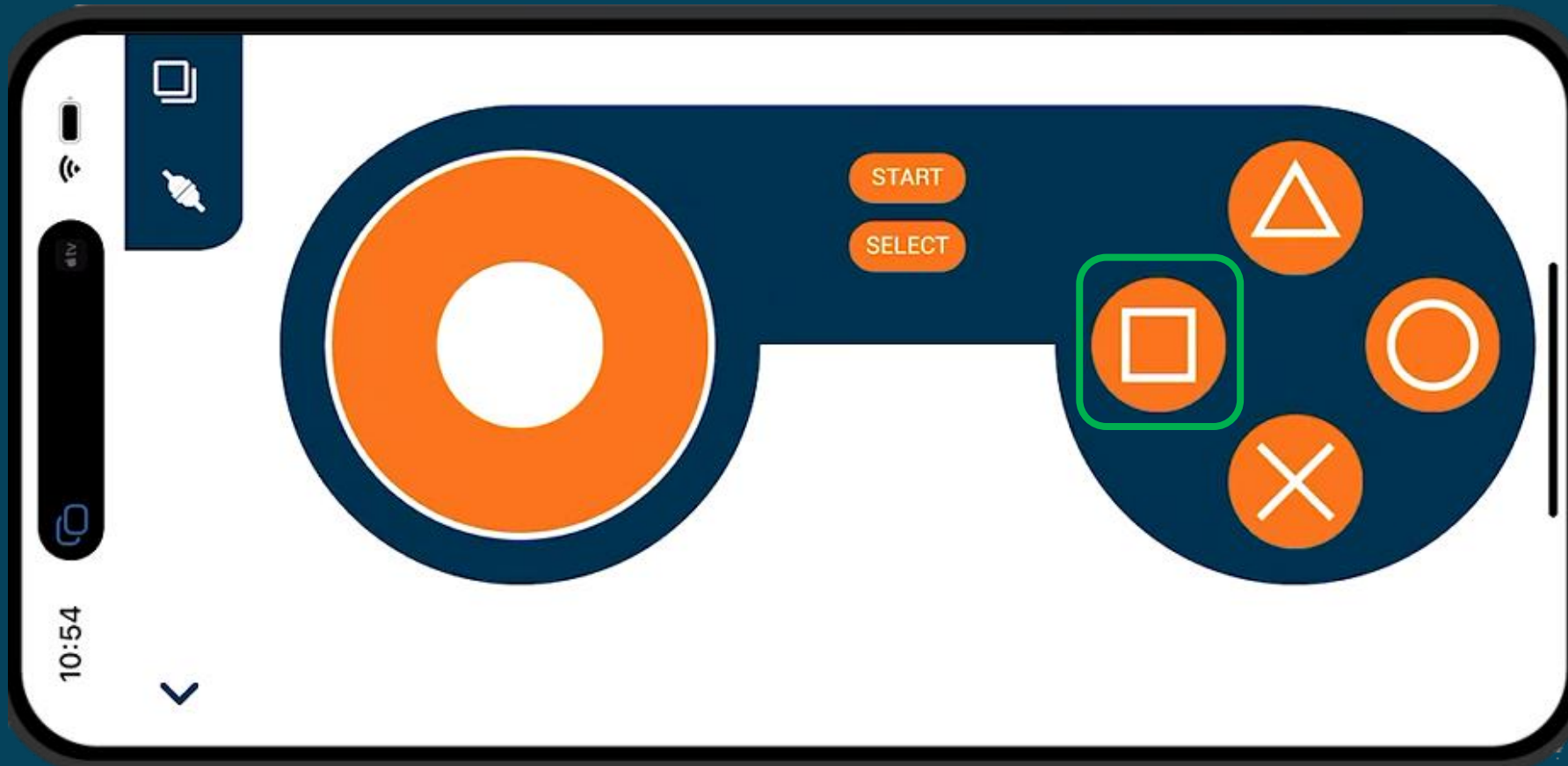
Use Dabble to operate the robot



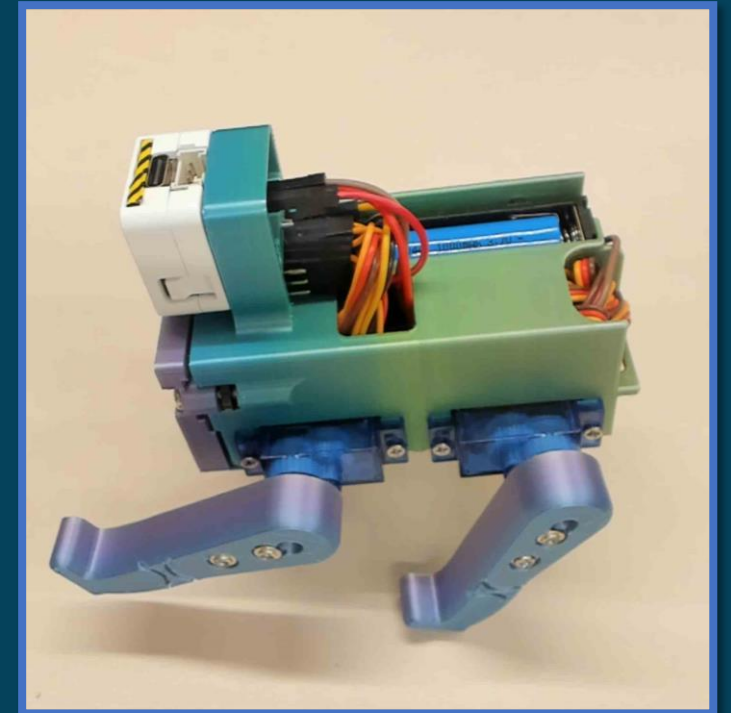
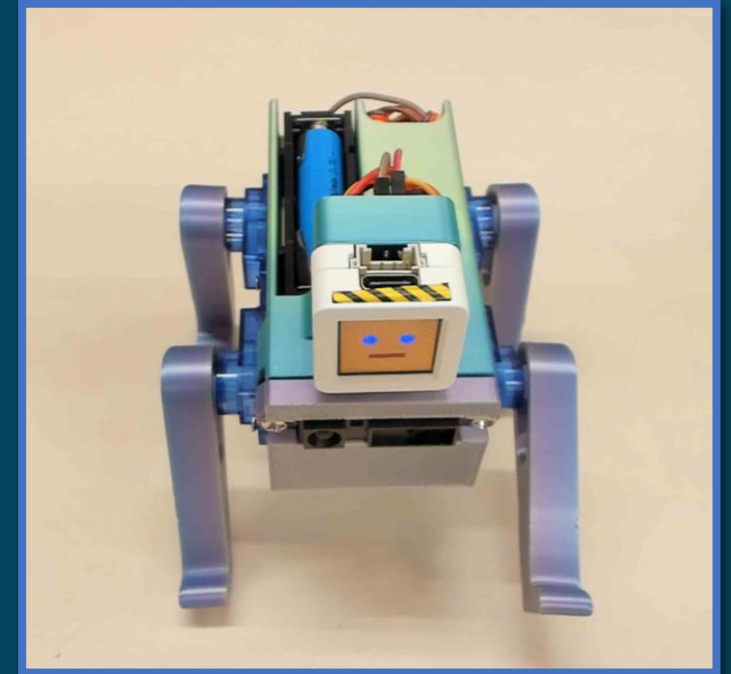
Right front leg up



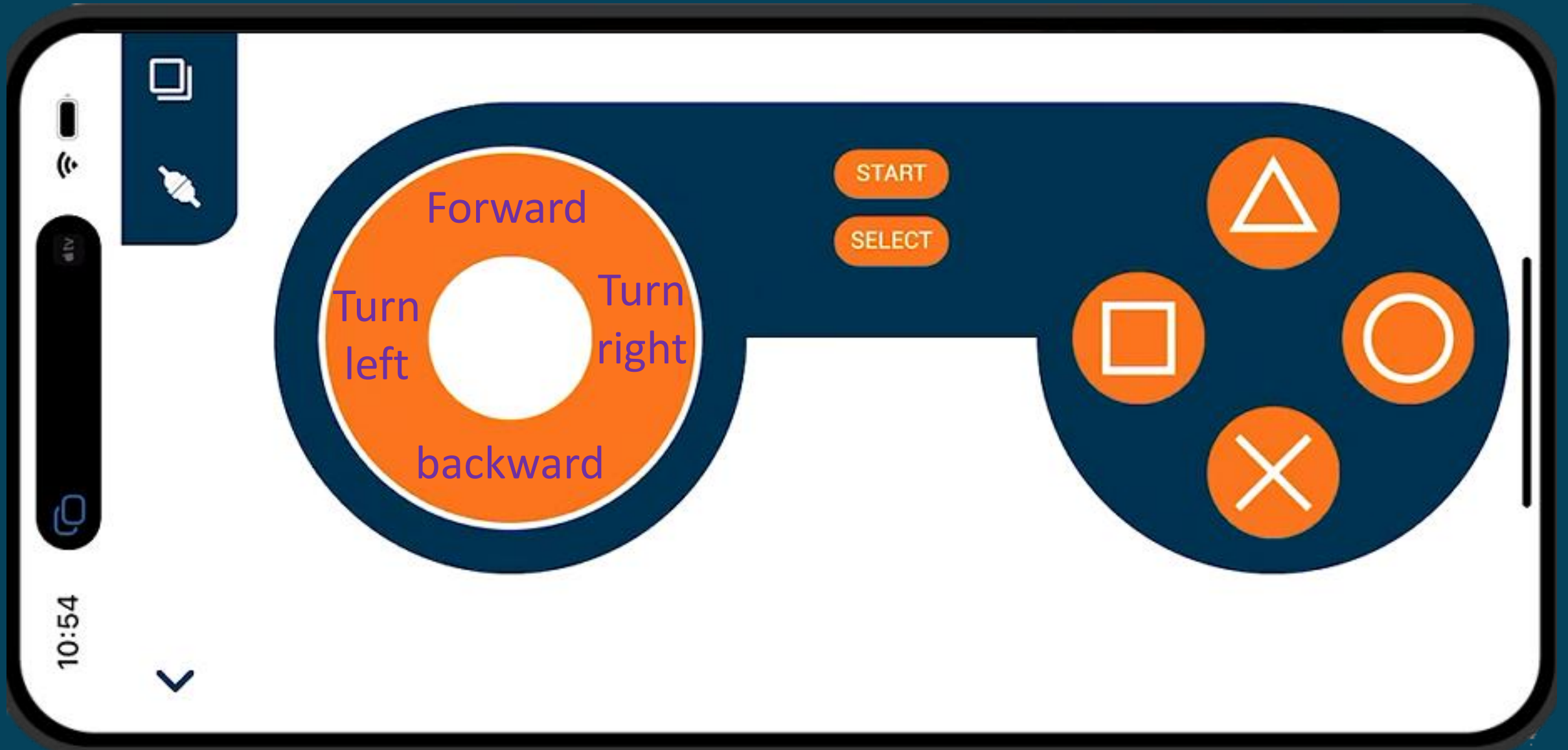
Use Dabble to operate the robot



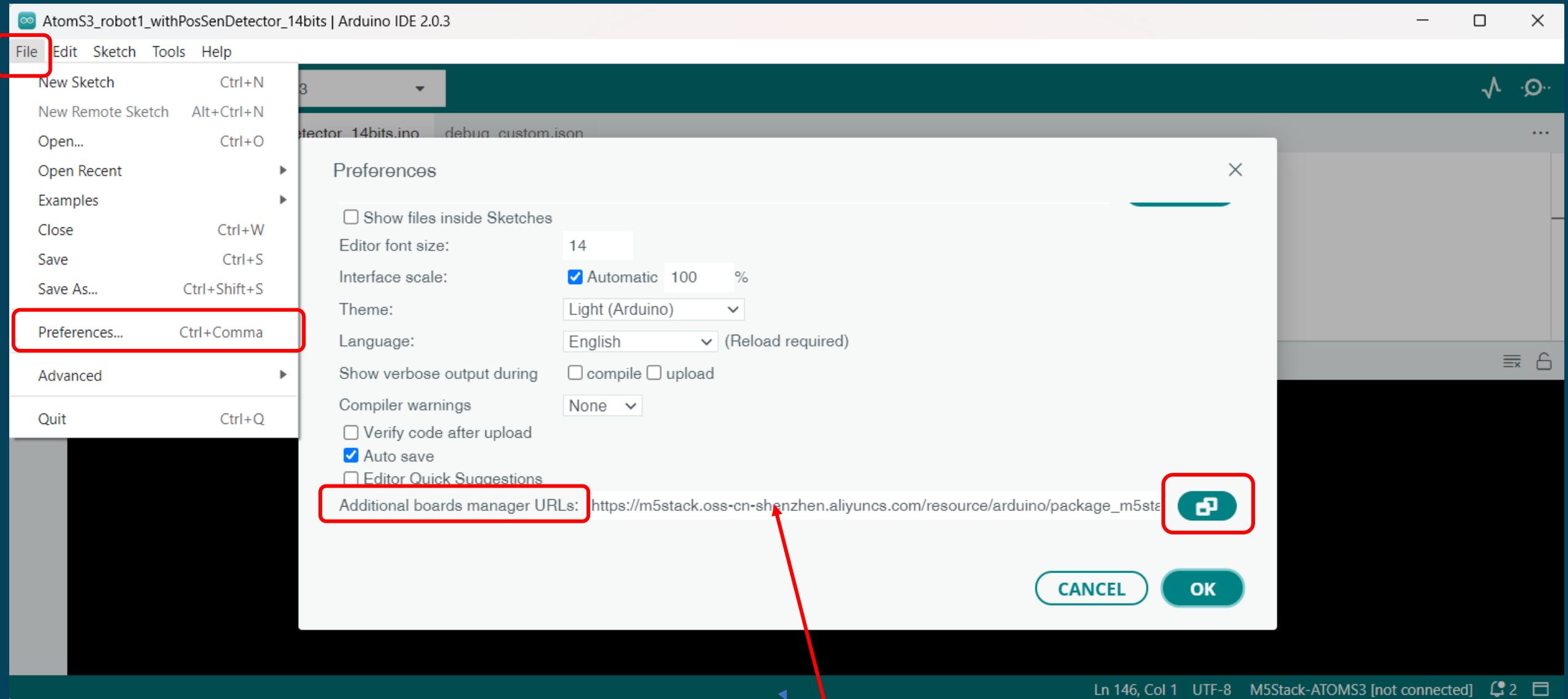
Left front leg up



Use Dabble to operate the robot (Walking)

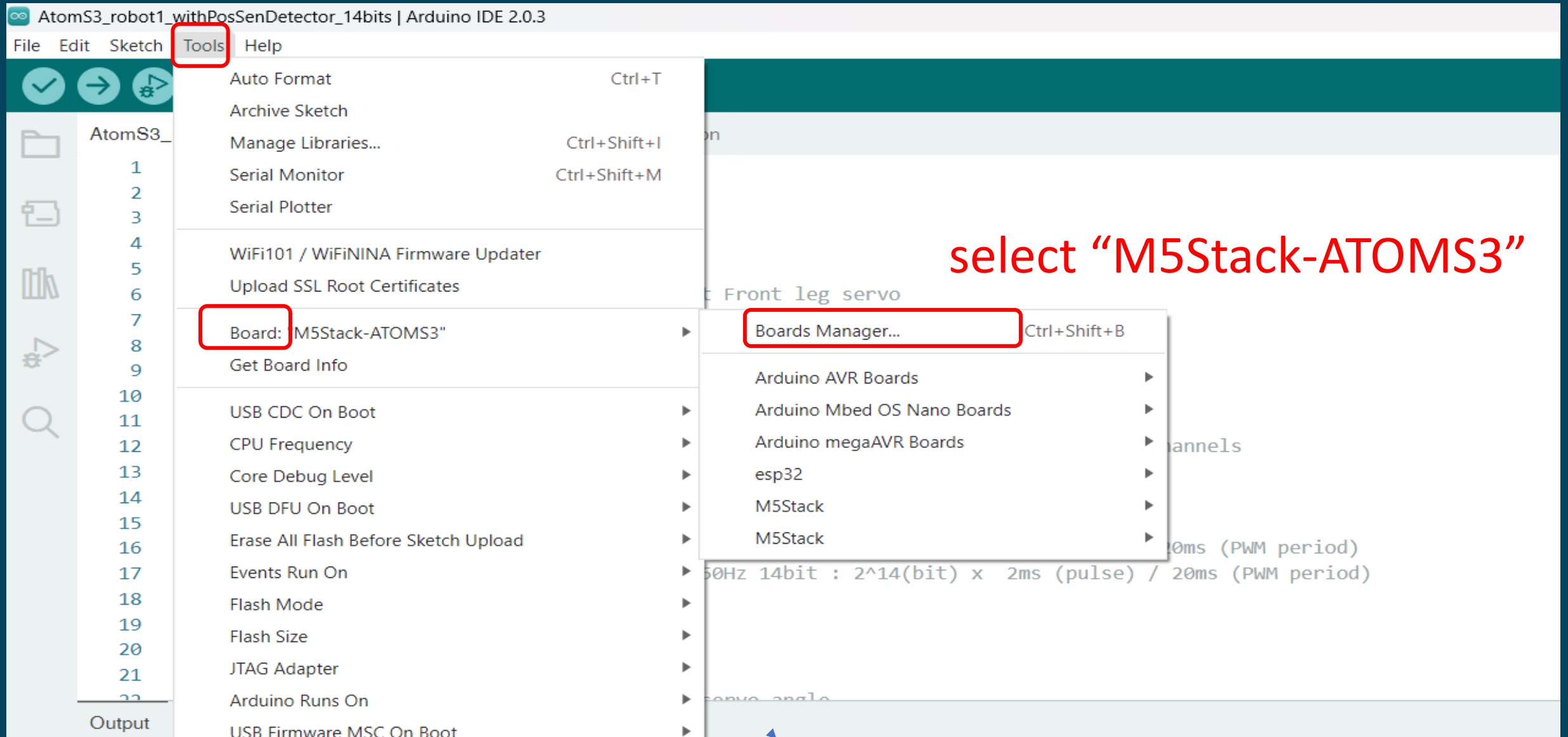


Coding with the Arduino IDE

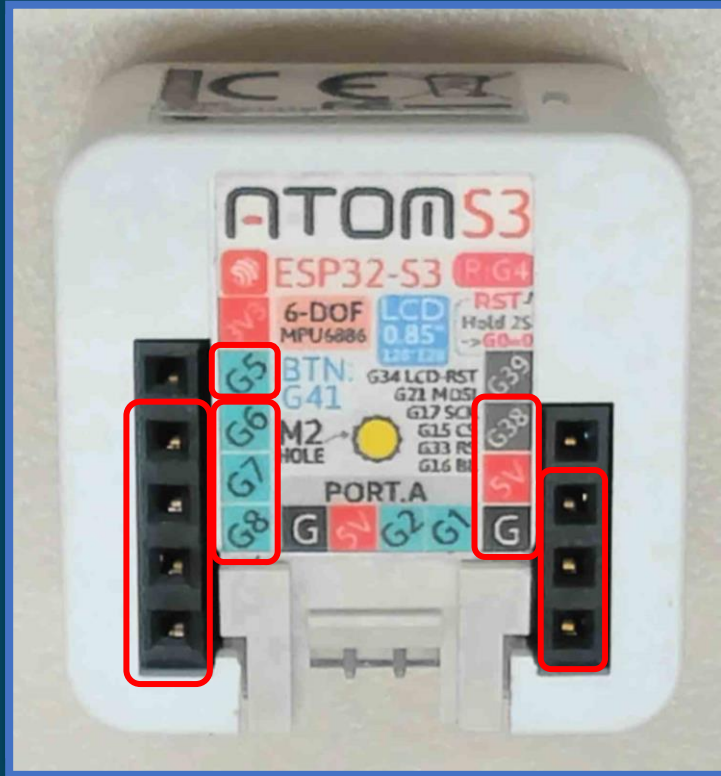


https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_index.json

Coding with the Arduino IDE

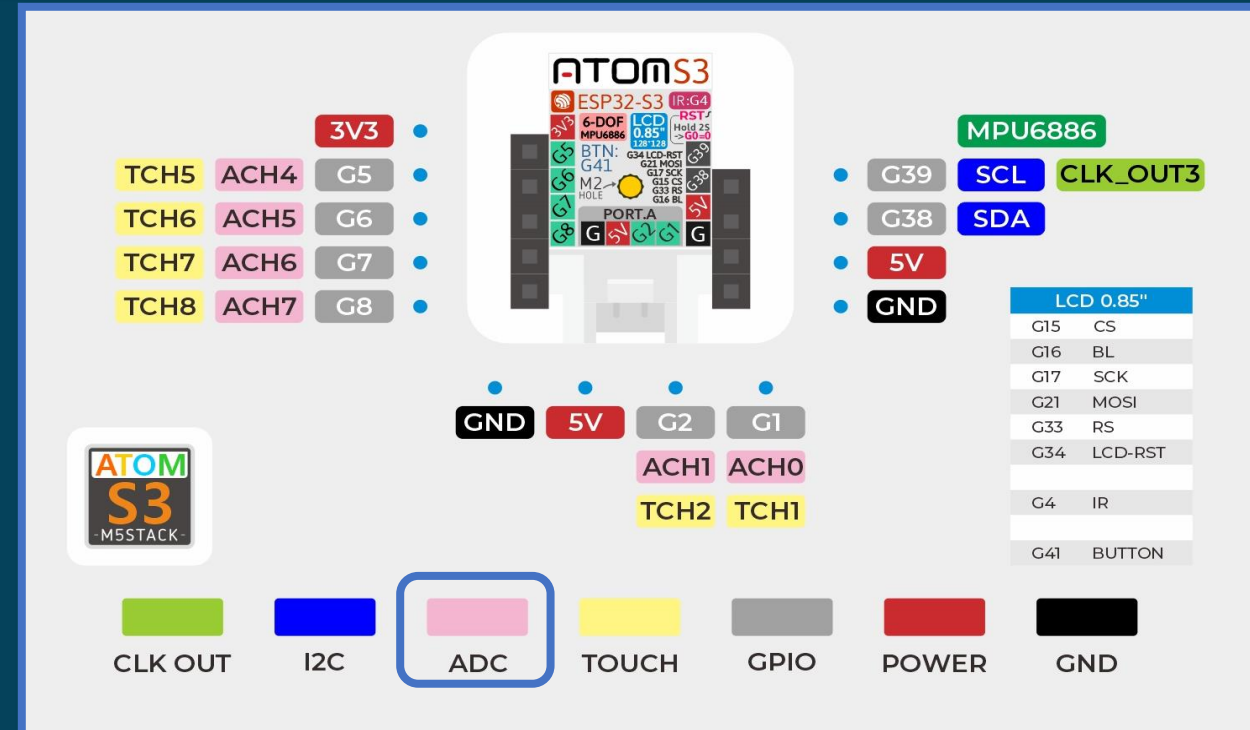


Coding with the Arduino IDE



```
4  #define Default_SETTINGS
5
6  const uint8_t Srv0 = 6; // GPIO pin 6 for Right Front leg servo
7  const uint8_t Srv1 = 7; // GPIO pin 7 for Right Back leg servo
8  const uint8_t Srv2 = 8; // GPIO pin 8 for Left Front leg servo
9  const uint8_t Srv3 = 38; // GPIO pin 38 for Left Back leg servo
10 const uint8_t Adc0 = 5; // GPIO pin 5: Analog to Digital converter port
11
```

Reserve G5 (ADC) for PSD sensor



Coding with the Arduino IDE

```
12  const uint8_t srv_CH0 = 0, srv_CH1 = 1, srv_CH2 = 2, srv_CH3 = 3; // define servo channels
13  const double PWM_Hz = 50;      // Pulse width modulation frequency => PWM period = 1/50 = 20ms
14  const uint8_t PWM_resolution = 14; // PWM resolution 14bit (0~16383)
15
16  int pulse_All_Left = 409;    // 0deg : 2^14(bit) x 0.5ms (pulse) / 20ms (PWM period)
17  int pulse_All_Right = 2048; //180deg: 2^14(bit) x 2.5ms (pulse) / 20ms (PWM period)
18
19  int all_left = 0;
20  int all_right = 180;
```

We can offer various analogue values to control the pulse width

Example:

Analogue values for **14-bit** pins can vary **from 0 to 16383**. ($2^{14} = 16,384$)

Servo Motor

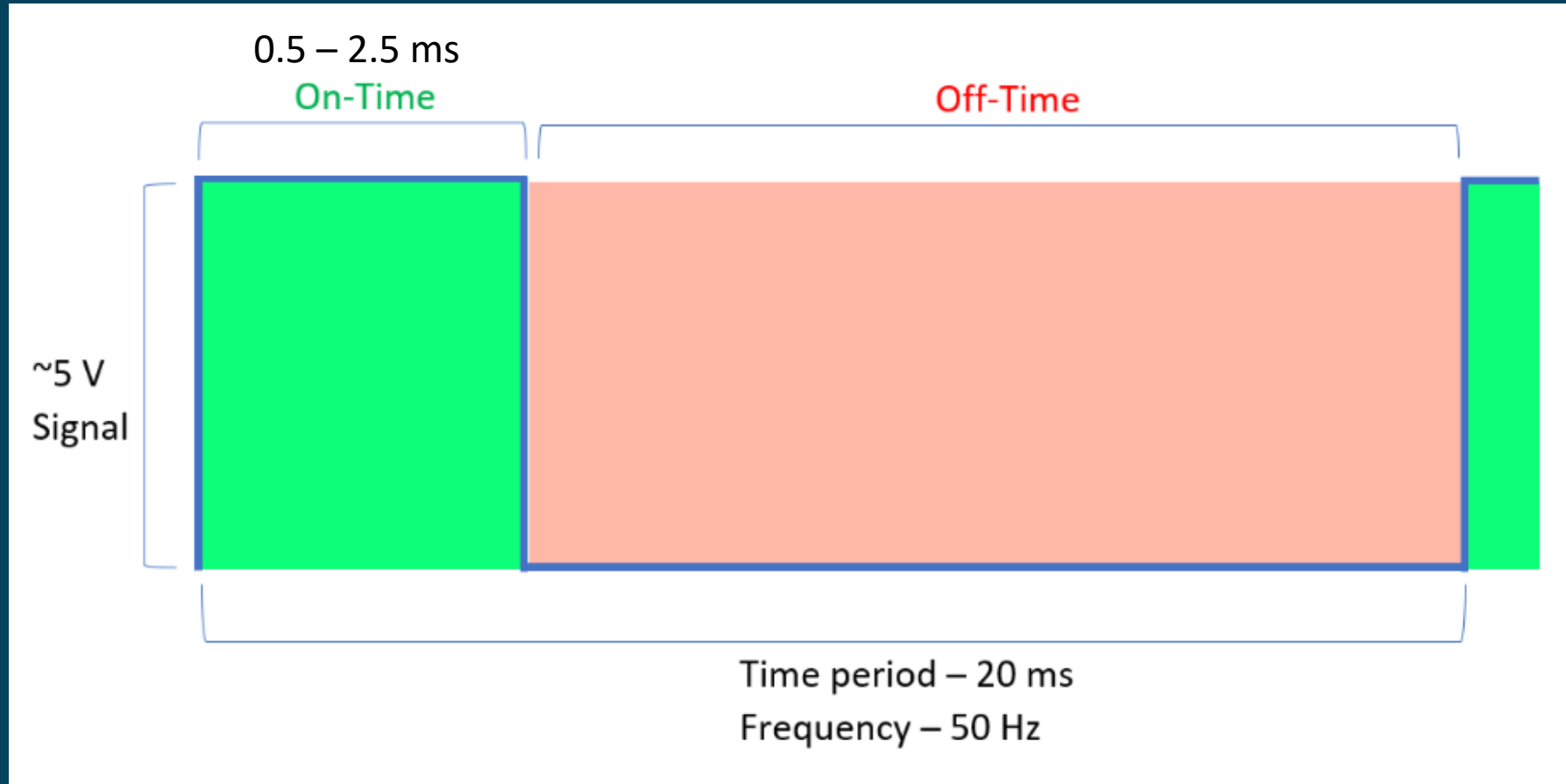
PWM frequency: 50 Hz

PWM period: $1/50 = 0.02\text{s}$
= 20ms

PWM resolution: 14bits

PWM pulse width:

0 – 16383 (2^{14})



You can rotate the servo motor shaft from 0° to 180° by varying the PWM pulse width from 0.5ms to 2.5ms.

$$0^\circ : 16384 \times \frac{0.5}{20} \approx 409$$

$$180^\circ : 16384 \times \frac{2.5}{20} \approx 2048$$

How to configure the program's parameters?



```
// Pulse width modulation frequency --> period 20ms  
const double PWM_Hz = 50;
```

```
//PWM resolution 14bit (0 ~ 16383)  $2^{14} = 16384$   
const uint8_t PWM_resolution = 14;
```

```
//  $2^{14}(\text{bit}) \times 0.5\text{ms} (\text{pulse}) / 20\text{ms} (\text{PWM period})$   
int pulse_All_Left = 409; // servo angle =  $0^\circ$ 
```

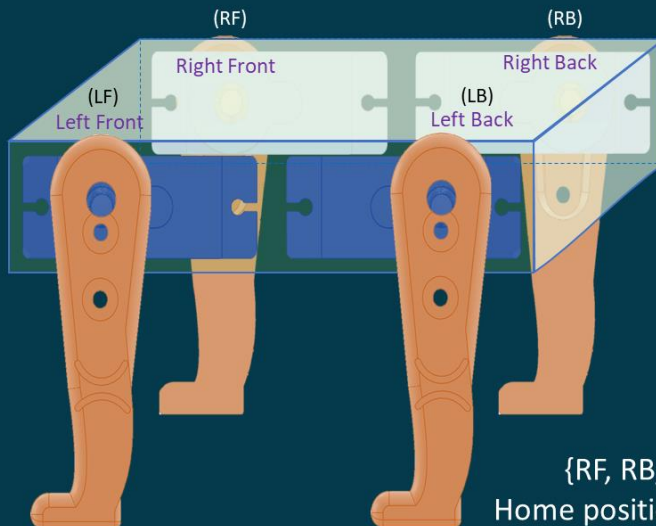
```
//  $2^{14}(\text{bit}) \times 2.5\text{ms} (\text{pulse}) / 20\text{ms} (\text{PWM period})$   
int pulse_All_Right = 2048;
```

```
int all_left = 0; //  $0^\circ$   
int all_right = 180; //  $180^\circ$ 
```

Coding with the Arduino IDE

```
22  int HomePosAng[] = {85,95,103,93}; // initial servo angle
23  int ang0[4]; // temporary variable
24  int ang1[4]; // temporary variable
25  float ts=120; // move on to the next step after 120ms
26  float td=10; // control the speed of the servo angle changes
27
28  int position_status = 0; // to prevent from 2 instructions run at the same time
29
```

Move forward from rest: 1



Home position:
{~90,~90,~90,~90}

{RF, RB, LF, LB}:
Home position + {0,0,0,0}

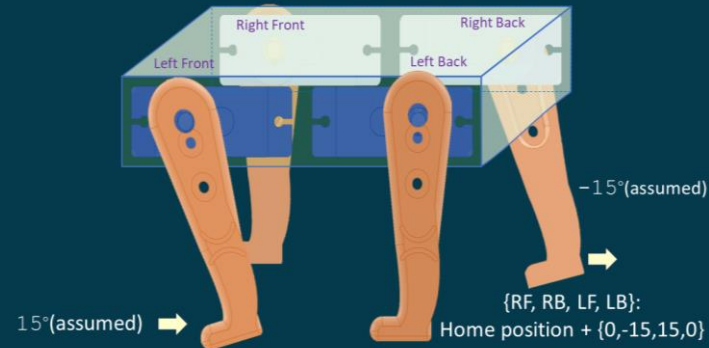
Calibration is required for the robot to stand properly

To alter the walking pace, modify these values

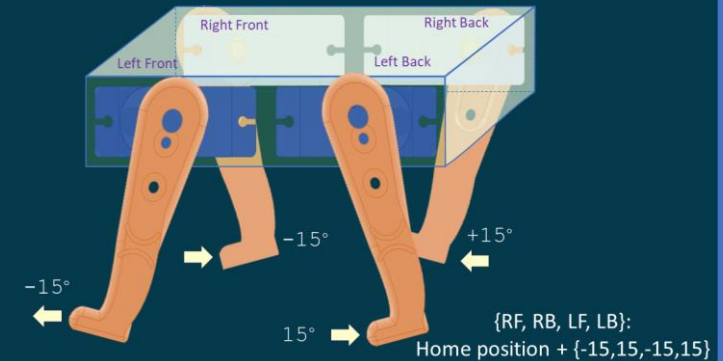
Coding with the Arduino IDE

```
30 // Forward Step
31 int f_s[4][4]={
32     {0,-15,15,0},
33     {-15,15,-15,15},
34     {-15,0,0,15},
35     {15,-15,15,-15}};
36
37 // Backward Step
38 int b_s[4][4]={
39     {0,15,-15,0},
40     {15,-15,15,-15},
41     {15,0,0,-15},
42     {-15,15,-15,15}};
43
44 // Left Turn_Step
45 int l_s[3][4]={
46     {-15,15,15,-15},
47     {-15,0,0,-15},
48     {0,0,0,0}};
```

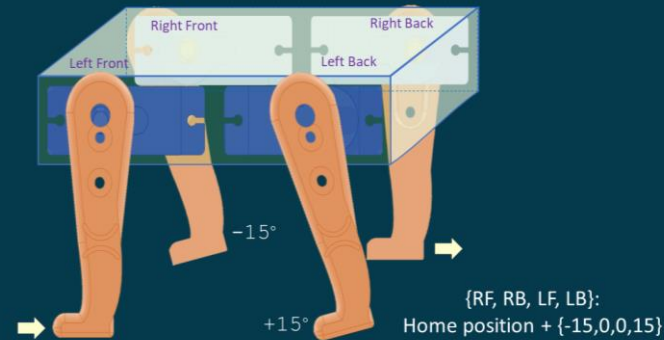
Move forward from rest: 2



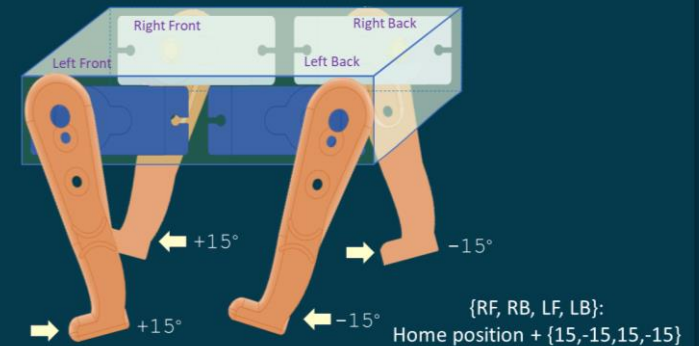
Move forward from rest: 3



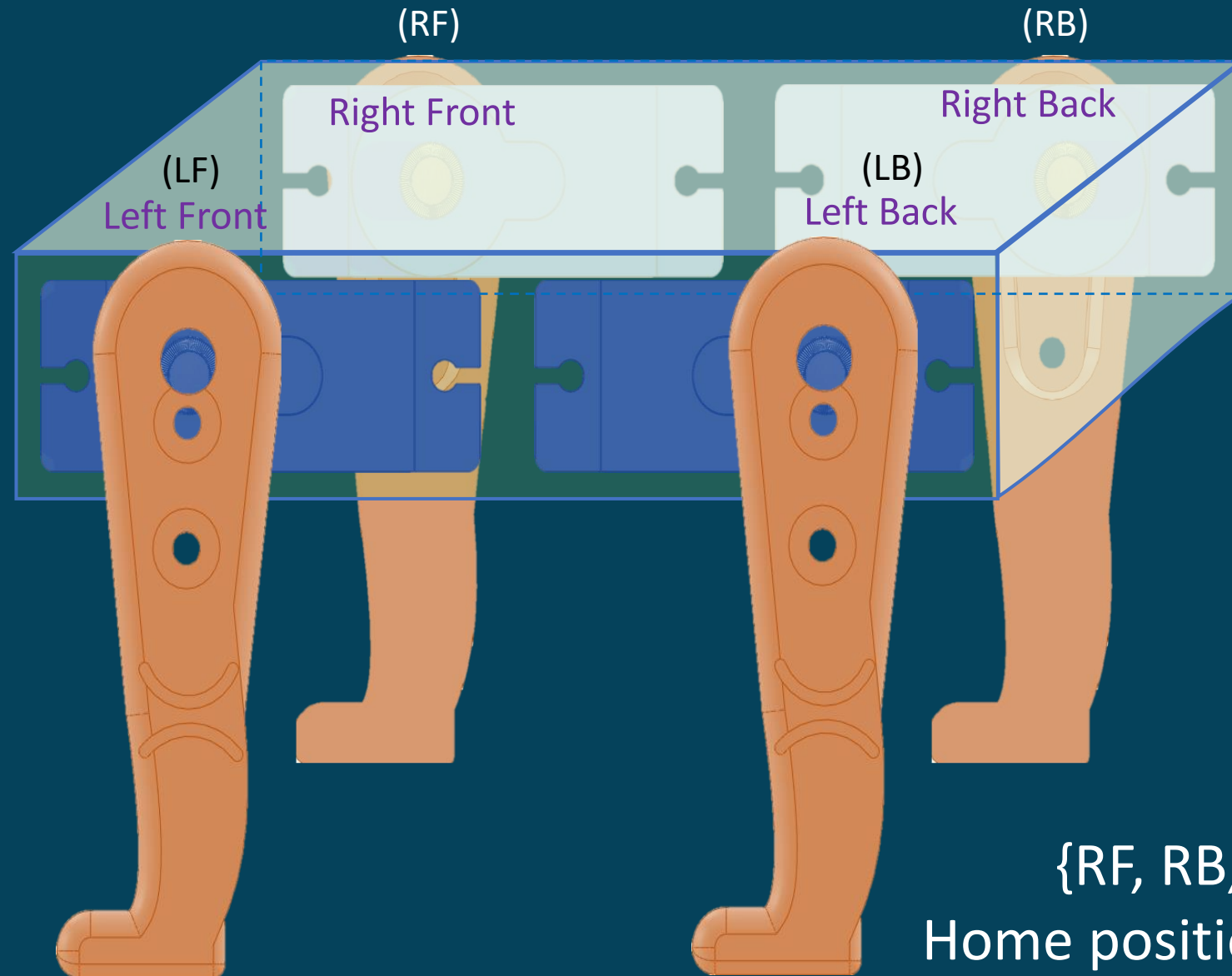
Move forward from rest: 4



Move forward from rest: 5



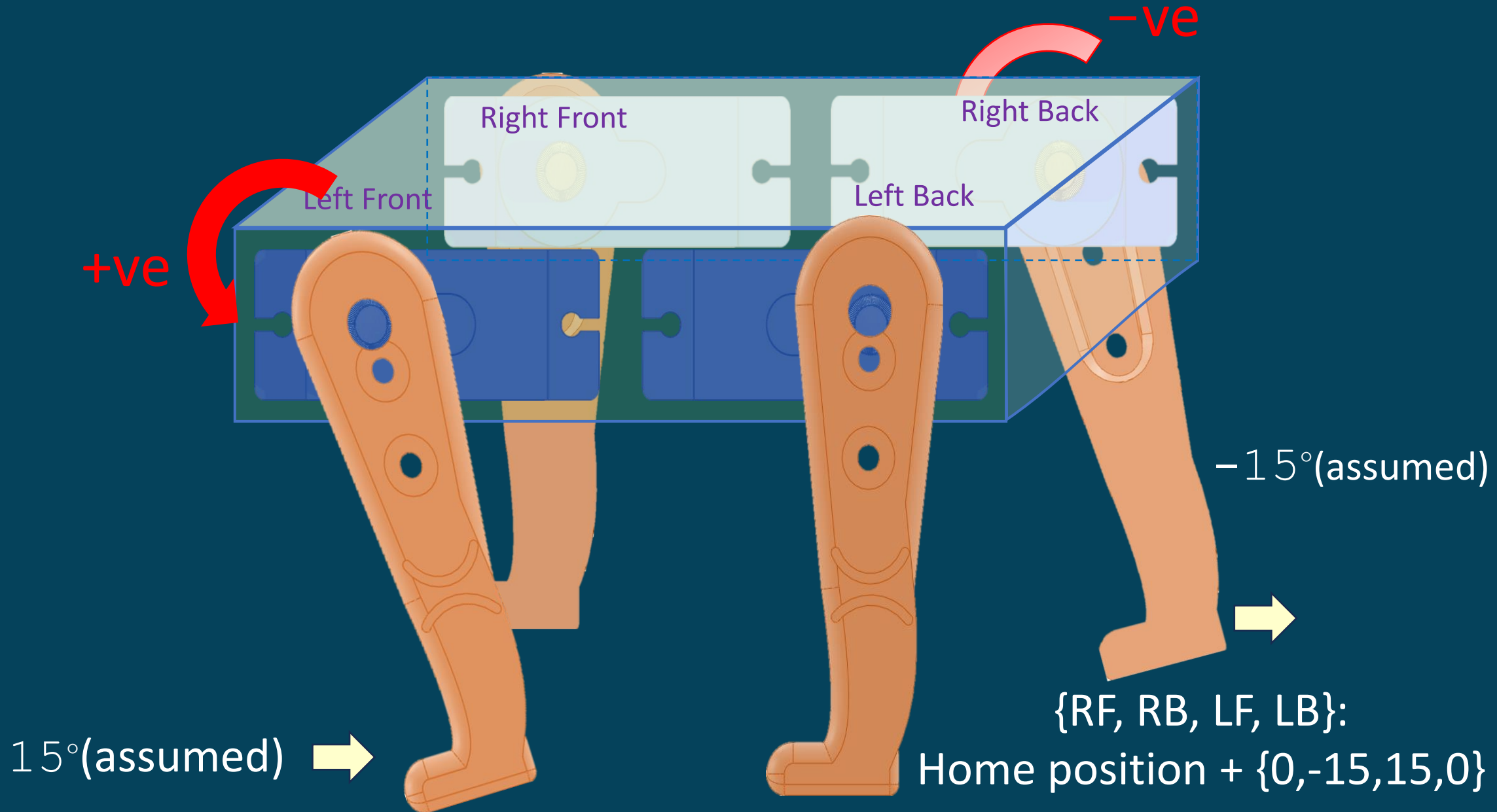
Move forward from rest: 1



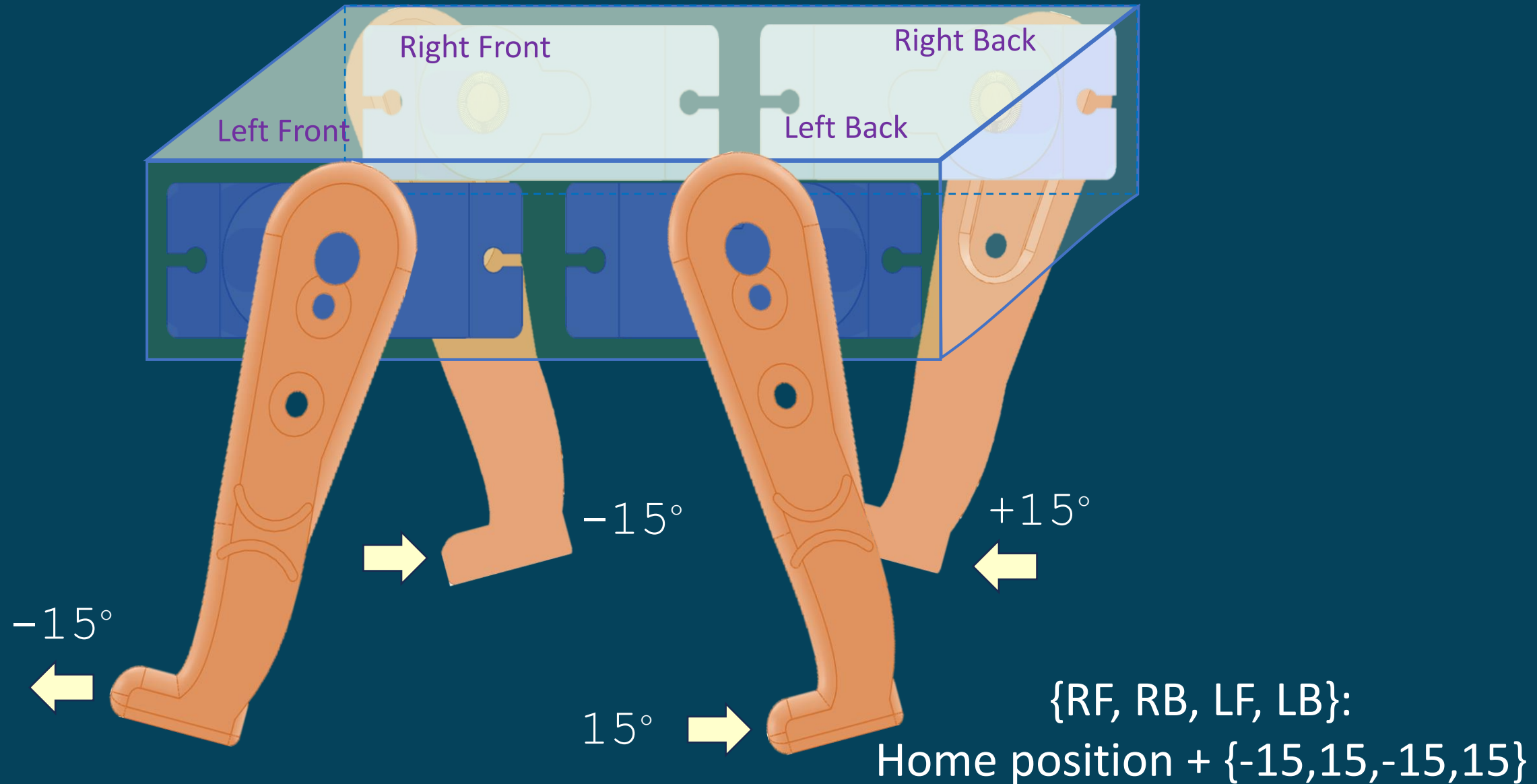
Home position:
 $\{\sim 90, \sim 90, \sim 90, \sim 90\}$

$\{RF, RB, LF, LB\}$:
Home position + $\{0, 0, 0, 0\}$

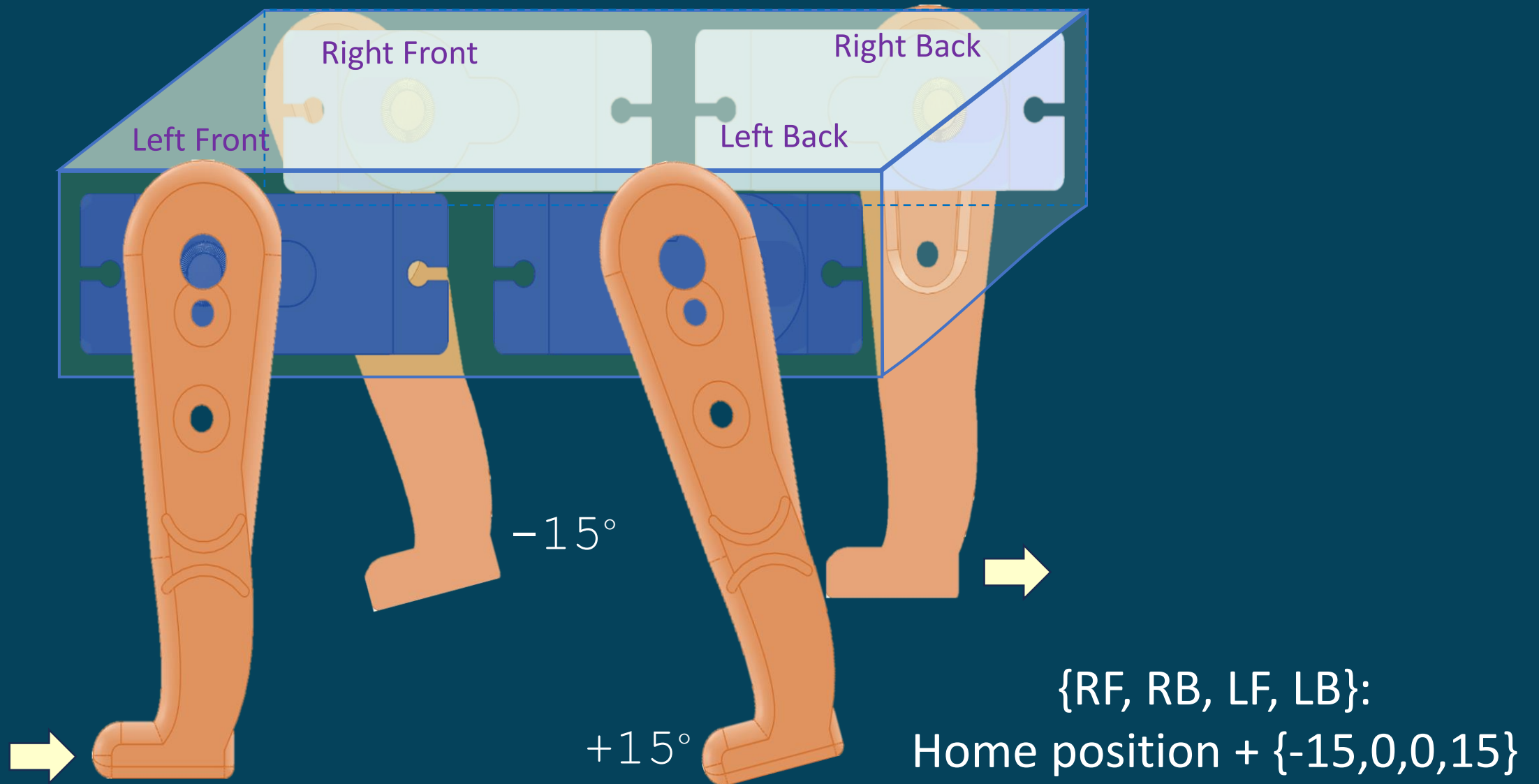
Move forward from rest: 2



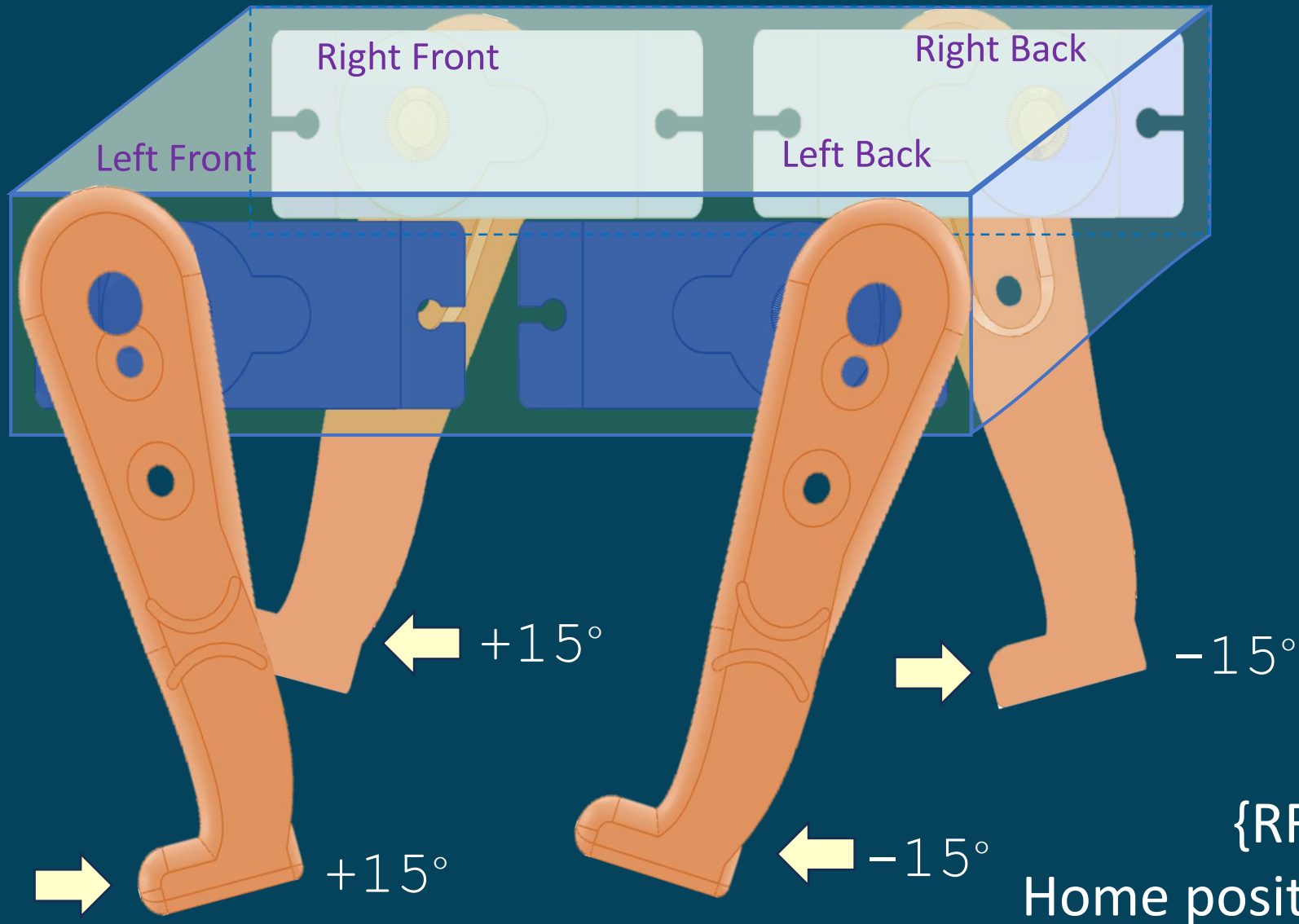
Move forward from rest: 3



Move forward from rest: 4

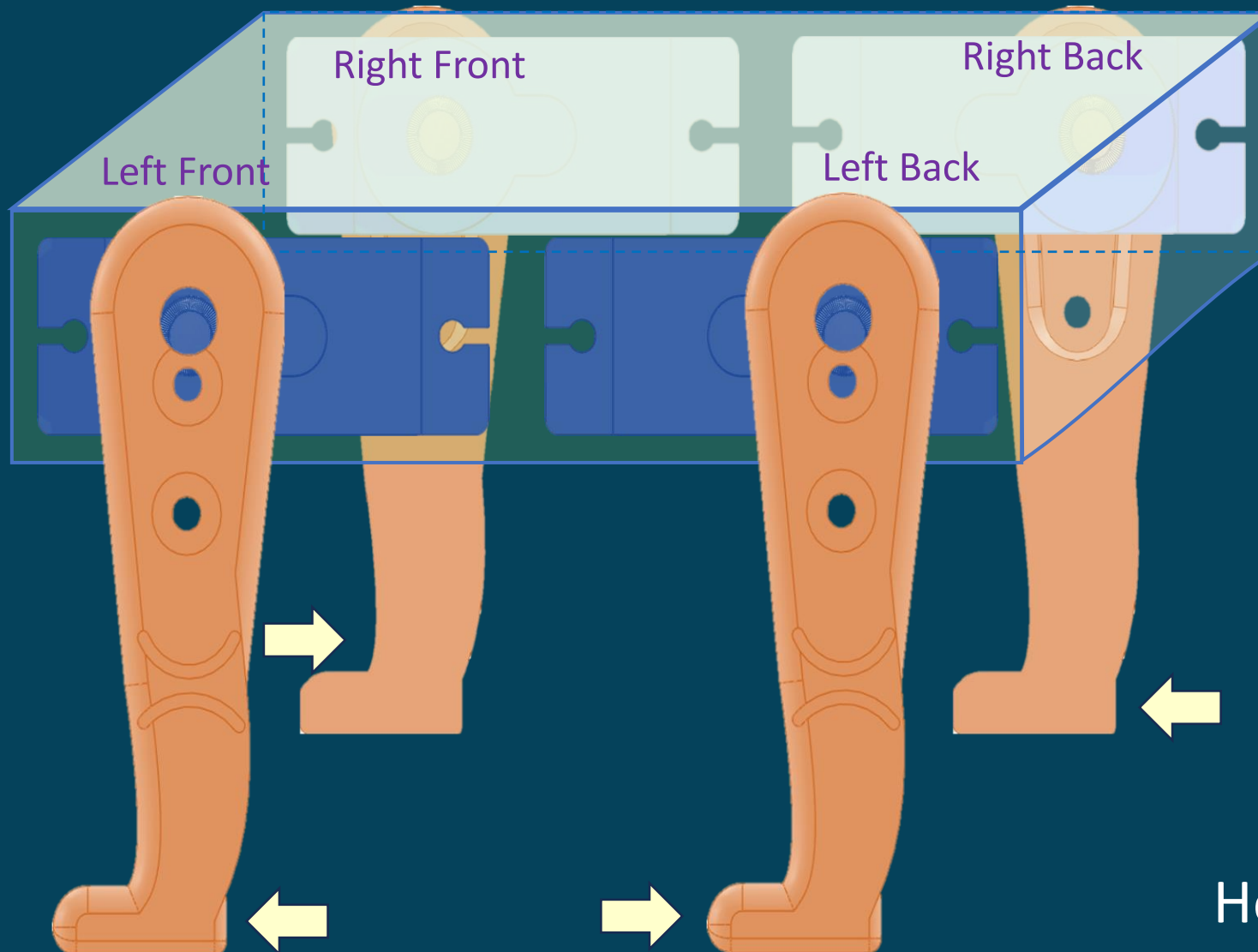


Move forward from rest: 5



{RF, RB, LF, LB}:
Home position + {15,-15,15,-15}

Move forward from rest: 6



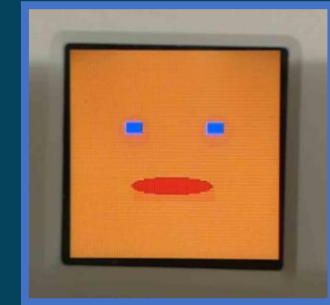
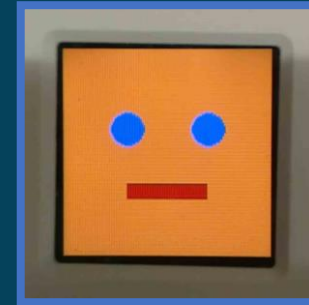
{RF, RB, LF, LB}:
Home position + {0,0,0,0}

Coding with the Arduino IDE

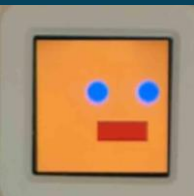
```
84 void face_default_eye(void *pvParameters){
85     while(1)
86     {
87         face_clear();
88         M5.Lcd.fillCircle(39,49,10,0x001F); // eye
89         M5.Lcd.fillCircle(89,49,10,0x001F); // eye
90         M5.Lcd.fillRect(39,83,50,10,0x8800); // mouth
91         delay(1000);
92         face_clear();
93         M5.Lcd.fillRect(34,44,10,7,0x001F); // eye
94         M5.Lcd.fillRect(84,44,10,7,0x001F); // eye
95         M5.Lcd.fillEllipse(62,83,25,5,0xC800); // mouth
96         delay(500);
97     }
98 }
```

```
100 void face_normal(){
101     face_clear();
102     M5.Lcd.fillCircle(39,49,10,0x001F);
103     M5.Lcd.fillCircle(89,49,10,0x001F);
104     M5.Lcd.fillRect(39,83,50,10,0x8800);
105 }
```

```
107 void face_changed(){
108     M5.Lcd.fillScreen(M5.Lcd.color565(255, 0, 0)); // Red background
109     M5.Lcd.fillCircle(39,49,10,0x001F);
110     M5.Lcd.fillCircle(89,49,10,0x001F);
111     M5.Lcd.fillRect(39,73,50,40,0xFFFF);
112 }
```



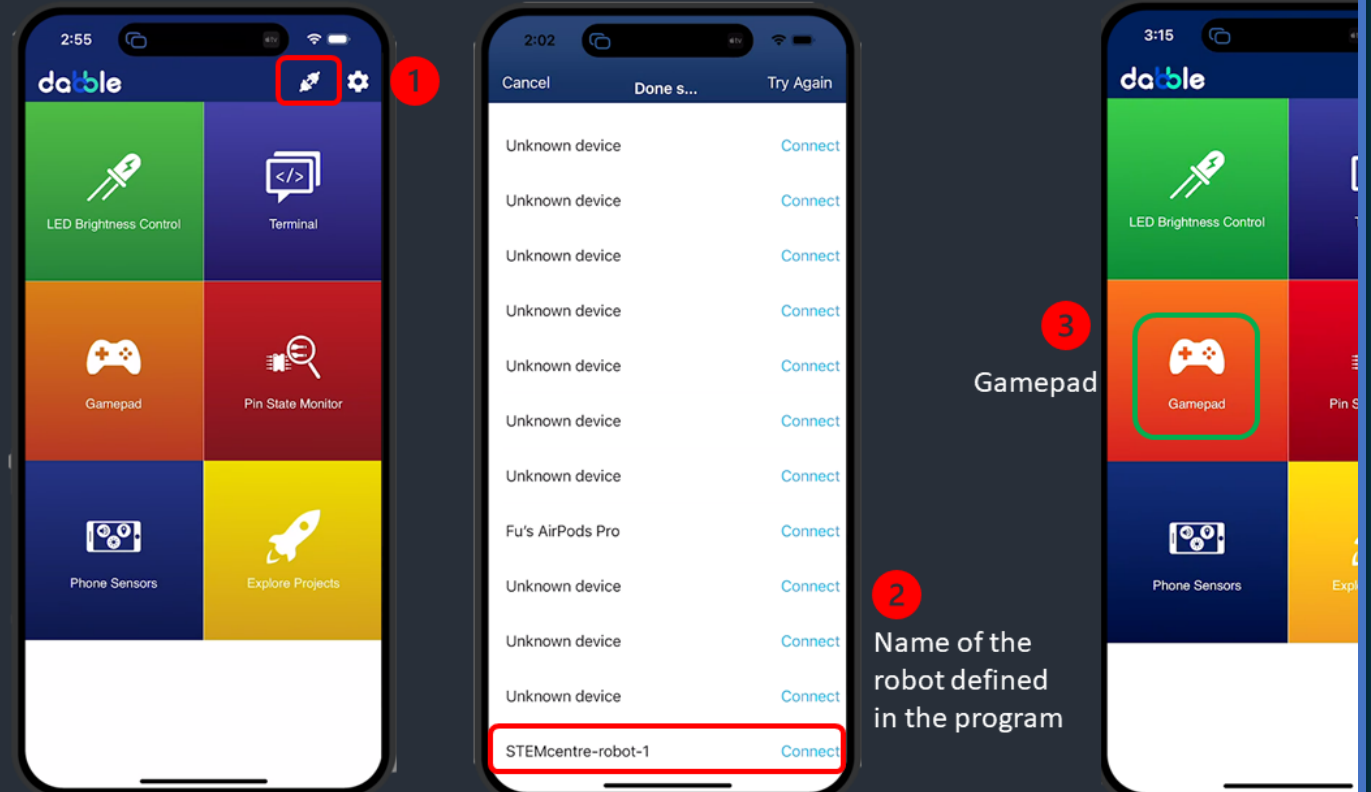
```
114 void look_right() {
115     face_clear();
116     M5.Lcd.fillCircle(19,49,10,0x001F);
117     M5.Lcd.fillCircle(69,49,10,0x001F);
118     M5.Lcd.fillRect(19,78,50,20,0x8800);
119 }
120
121 void look_left() {
122     face_clear();
123     M5.Lcd.fillCircle(59,49,10,0x001F);
124     M5.Lcd.fillCircle(109,49,10,0x001F);
125     M5.Lcd.fillRect(59,78,50,20,0x8800);
126 }
```



Coding with the Arduino IDE

```
299 void setup() {
300     Serial.begin(151200);
301     M5.begin(true, true, false, false); // Initialise M5AtomS3 (LCD, USB serial, I2C(38,39), LED)
302     M5.Lcd.setRotation(4); // Rotate the screen 90 deg in counterclockwise dir
303     Dabble.begin("STEMcentre-robot-1"); // set bluetooth name for connection
304
305     pinMode(Srv0, OUTPUT);
306     pinMode(Srv1, OUTPUT);
307     pinMode(Srv2, OUTPUT);
308     pinMode(Srv3, OUTPUT);
309     pinMode(Adc0, INPUT);
310
311     // configure servo PWM channel and frequency
312     ledcSetup(srv_CH0, PWM_Hz, PWM_resolution);
313     ledcSetup(srv_CH1, PWM_Hz, PWM_resolution);
314     ledcSetup(srv_CH2, PWM_Hz, PWM_resolution);
315     ledcSetup(srv_CH3, PWM_Hz, PWM_resolution);
316
317     // Configure servo pins and channels
318     ledcAttachPin(Srv0, srv_CH0);
319     ledcAttachPin(Srv1, srv_CH1);
320     ledcAttachPin(Srv2, srv_CH2);
321     ledcAttachPin(Srv3, srv_CH3);
322
323     face_normal();
324
325     Initial_setting();
```

Establish a connection between Dabble and the robot



Coding with the Arduino IDE

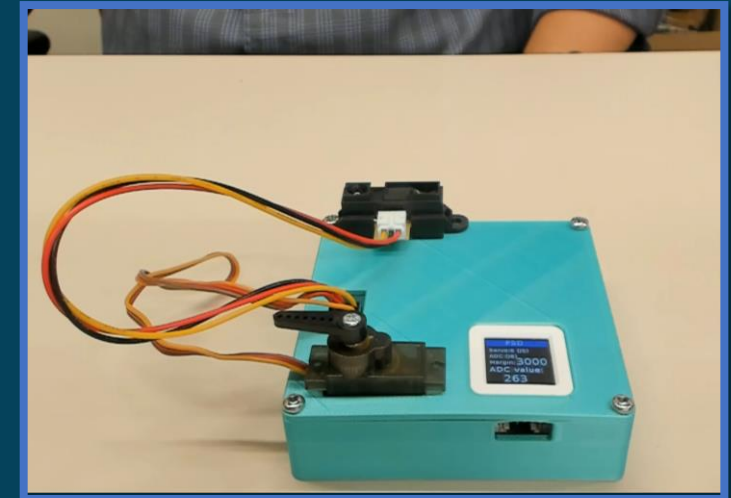
```
292 float psd()  
293 {  
294     float volts = analogRead(Adc0)*5/4096;  
295  
296     return analogRead(Adc0);  
297 }
```



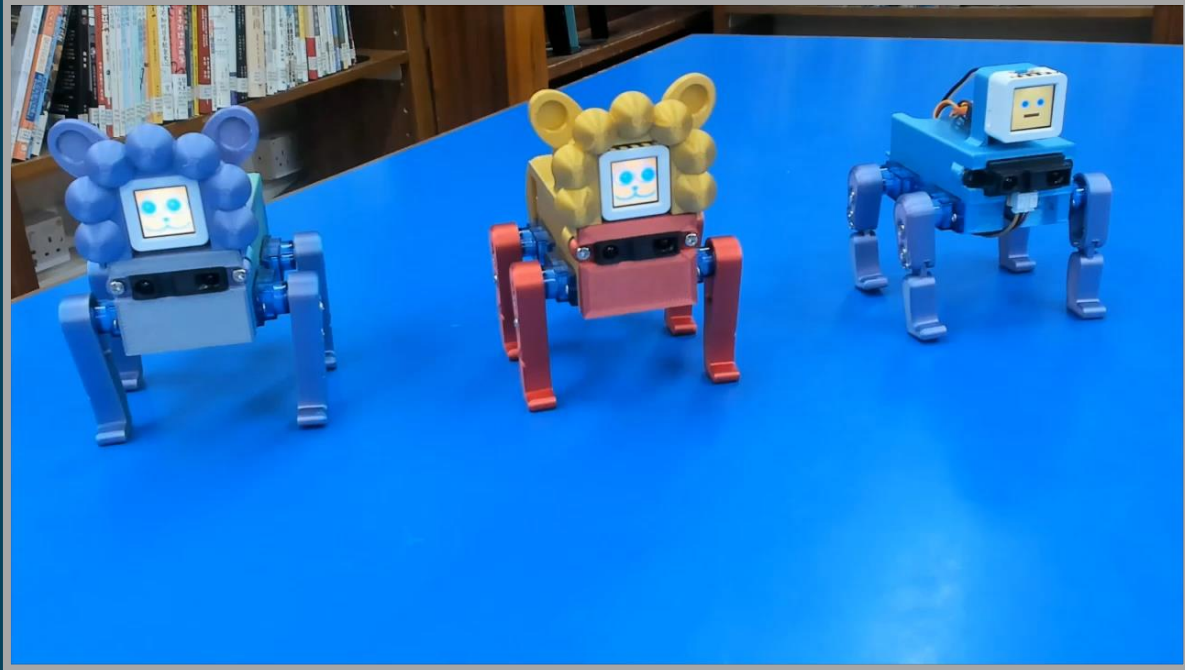
The Adc0 values from position sensitive detector range from 0 to 4095 (2^{12})

```
378 int virtual_distance = 0;  
379 virtual_distance = psd();  
380 USBSerial.println(psd());  
381 if(((psd())>3400)&&(psd())<4095) && (position_status == 0))  
382 {  
383     back_step();  
384     back_step();  
385 }  
386 if(((psd())>1800)&&(psd())<3400) && (position_status == 0))  
387 {  
388     forward_step();  
389 }  
390 }
```

Modify these settings to alter the robot's moving behaviour



Clothing design for the robot



You may design a robot uniform and print it using a 3D printer

Maintenance of the battery



To recharge the battery, use a smart universal charger suitable for 3.7V Lithium AAA Battery

Other references

