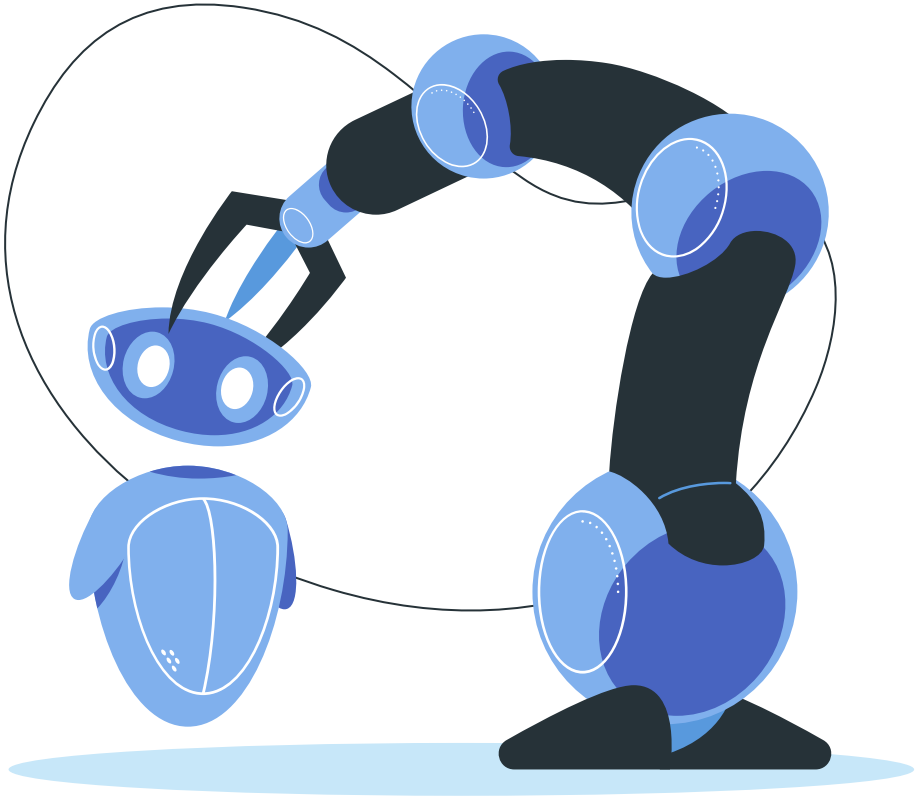
Me Arm

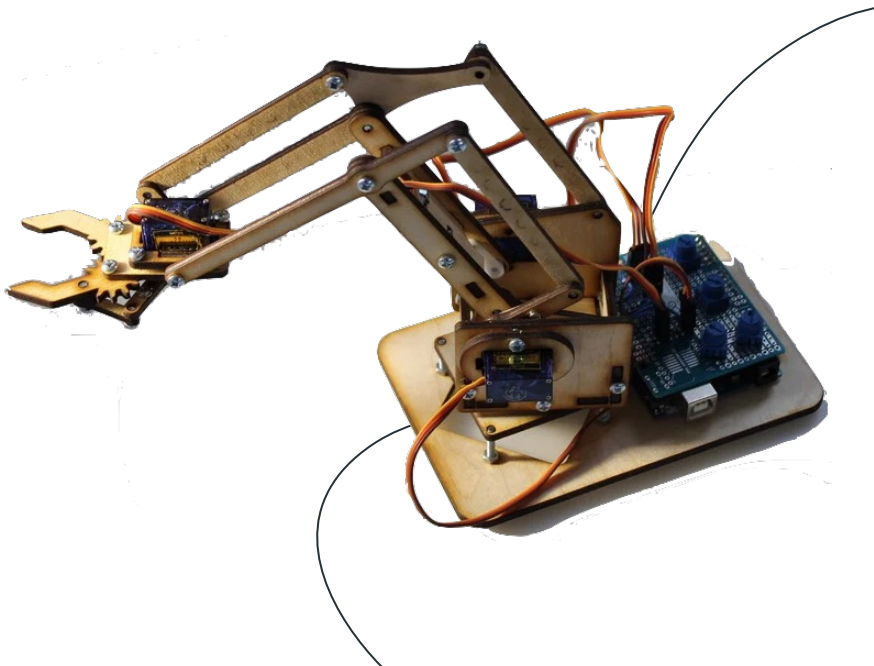
A robot with ROS and Arduino

Alessia Ture



Introduction to MeArm

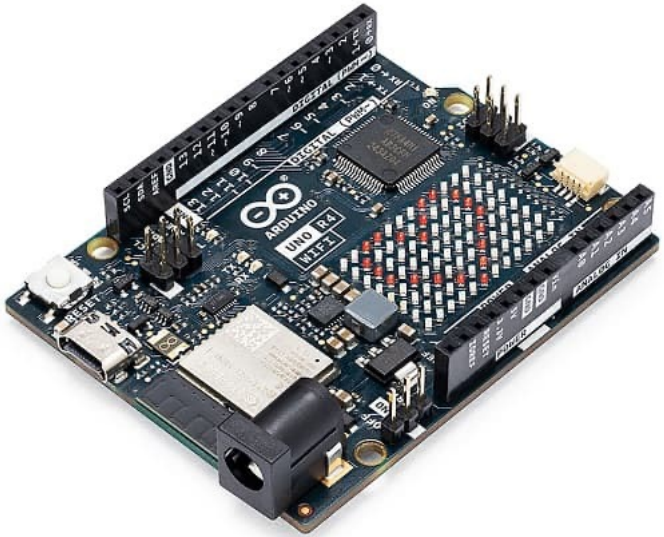
- **Compact and Open-Source**: The MeArm is a compact, open-source robotic arm known for its simplicity and accessibility, making it an ideal platform for educational purposes and hobbyist projects.
- **4DOF Architecture**: Featuring four degrees of freedom (DOF) powered by four micro servomotors
- **Project Objective:** control the MeArm using the Robot Operating System (ROS)



Arduino

Arduino is an open-source **microcontroller**

Arduino and ROS integration allows you to leverage the simplicity of Arduino for hardware control and the power of ROS for high-level communication and processing.



rosserial: what is it?

rosserial_client

Libraries for various microcontrollers that implement the rosserial communication protocol.

rosserial_server:

A ROS node that communicates with the rosserial_client and acts as a bridge between the microcontroller and the ROS ecosystem.

How integrate ROS & Arduino?

1

Arduino Node

rosserial_arduino library in sketch, allows the Arduino to publish and subscribe to ROS topics, call or provide ROS services, and use ROS time

2

Ros Host

run a node provided by rosserial, such as rosserial_python. This node opens a serial port connection to the Arduino. It then serializes ROS messages into a byte stream

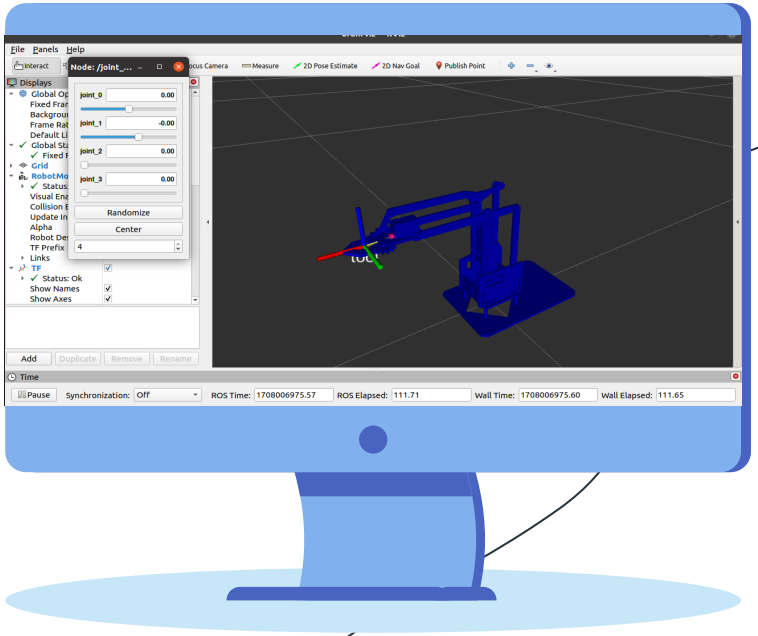
3

Connection

When the Arduino wants to publish data, rosserial_client packages the data into packets and sends them over the serial link. The rosserial_server receives these packets, unpacks them, and publishes

RViz

RViz (ROS Visualization) is an interactive 3D visualization tool for the Robot Operating System (ROS) that allows users to visualize robot architecture and sensor data

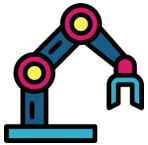


Arduino Node

- Arduino sketch allows the MeArm to receive position commands for its joints from a ROS node, moving the servomotors to specified degrees.
- Sketch subscribes to the **servo_pose_server** topic, which should receive messages like **mearm_model::ServoAngles**.
- A callback function, **servoAnglesCallback**, is defined which is automatically called every time a new message arrives on the topic. The function reads the angles from the messages and sets the servo motors corresponding to those angles

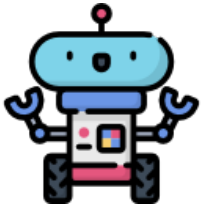
```
#include <Servo.h>
#include <ros.h>
#include <mearm_model/ServoAngles.h>

Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;

ros::NodeHandle nh;

void servoAnglesCallback(const mearm_model::ServoAngles& angles_msg) {
  servo1.write(angles_msg.servo1);
  servo2.write(angles_msg.servo2);
  servo3.write(angles_msg.servo3);
  servo4.write(angles_msg.servo4);
}

ros::Subscriber<mearm_model::ServoAngles> servoAnglesSub("servo_pose_server", &servoAnglesCallback);

void setup() {
  nh.getHardware()->setBaud(115200);
  nh.initNode();
  nh.subscribe(servoAnglesSub);

  servo1.attach(9);
  servo2.attach(10);
  servo3.attach(11);
  servo4.attach(12);
}

void loop() {
  nh.spinOnce();
}
```

Nodes



robot_joint_mover

Defines a service that allows users to set the position of the robot's joints by sending the name of a predefined set of positions. When the service receives a request, it looks for the corresponding set of locations in the **ROS Parameter Server.**



mearm_controller

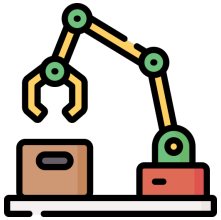
Creates a publisher on a topic called **servo_pose_server** that publishes messages of type **ServoAngles**. These messages contain the angles for the servo motors that the MeArm will use to position its joints

Messagges

Used for communication between nodes in a ROS system. This message is specifically designed to carry information about servomotor angles.

```
int16 servo1
int16 servo2
int16 servo3
int16 servo4
```

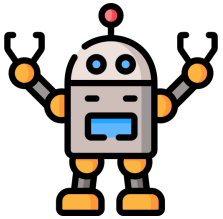
Topics



/servo_pose_server:

Message Type:
mearm_model/ServoAngles

Function: Used to send precise positioning commands to the MeArm's servo motors. Nodes can post to this topic to change joint positions.



/joint_states:

Message Type:
sensor_msgs/JointState

Function: Transmits the current state of the MeArm joints, including position, velocity and effort for each joint

Services

ROS node (**robot_joint_mover**) calls this service and sends the name of a set of locations through the *set_name* field.

The node, which manages the service, receives this request, executes the necessary logic to change the position of the and then would respond with **success = true** if the process is completed correctly, or **success = false** if there was an error or the set was not applicable.

```
string set_name
---
bool success
```

Parameter Server

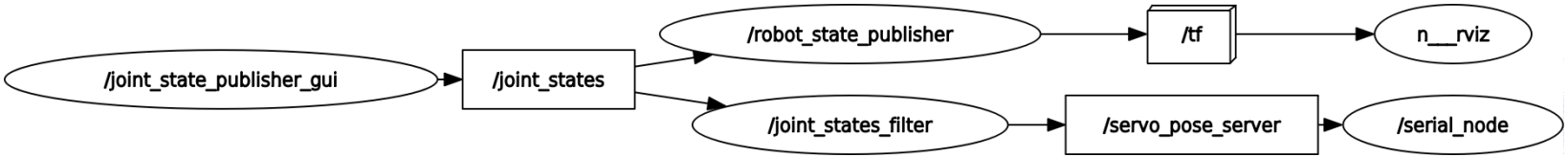
It is used to group the different positions that can be applied to the joints of the MeArm

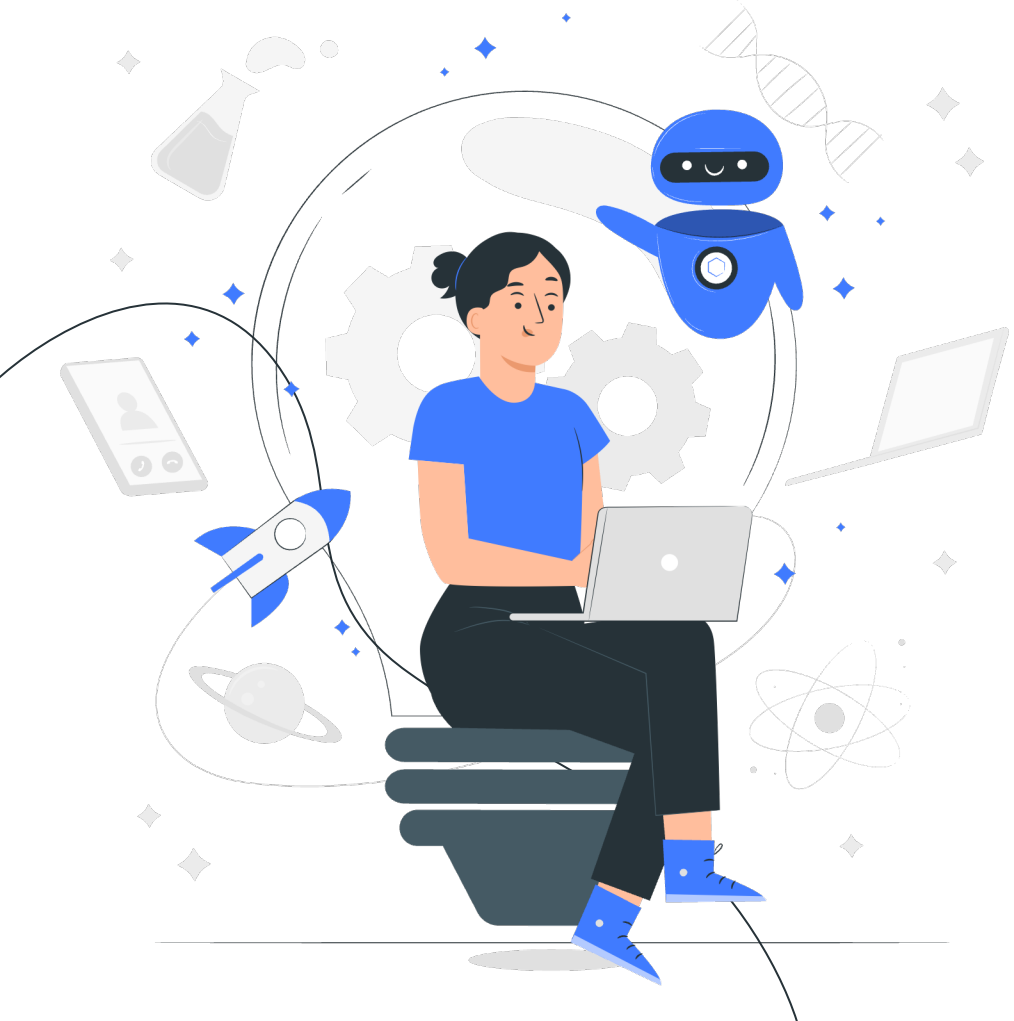
These sets of positions can be used to quickly command the robot to assume specific positions.

```
position_sets:
set1: [0.0, 0.5, -0.5, 0.25]
set2: [0.25, -0.5, 0.5, 0.0]
set3: [-0.25, 0.0, 0.25, 0.5]
```

System Architecture

The schema present topics and node involed in RViz-arduino communication





Thanks!

All the code show can be found at:
https://github.com/a-ture/MeArm-Robot/