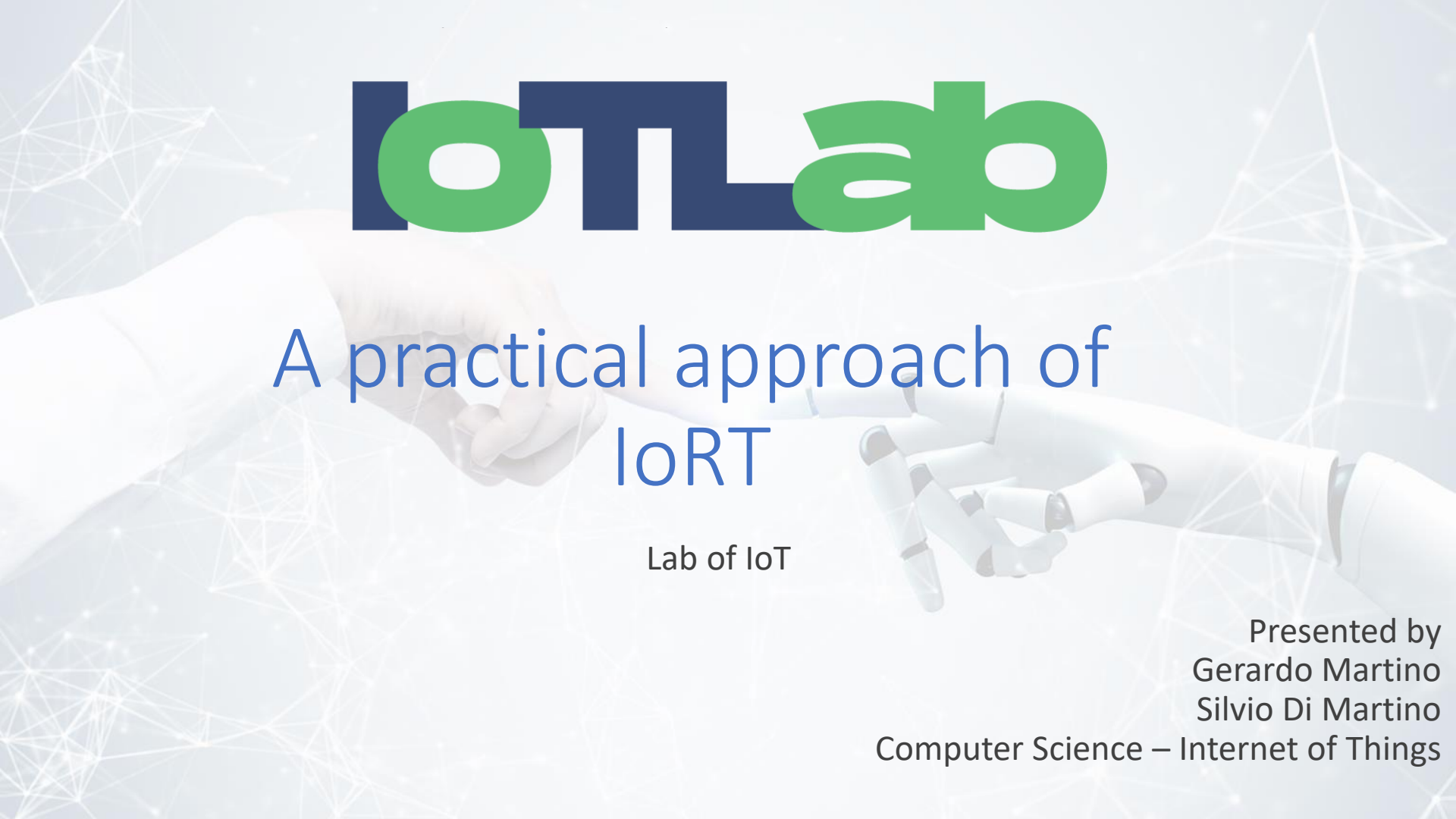
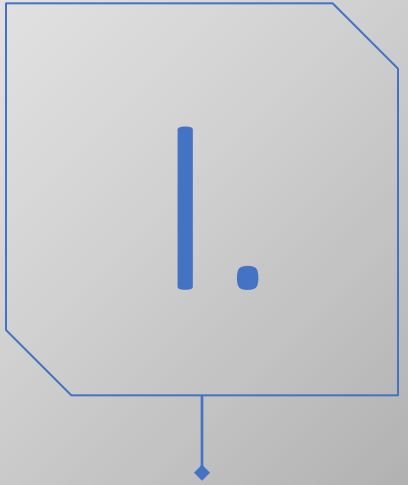


A practical approach of IoRT

Lab of IoT

Presented by
Gerardo Martino
Silvio Di Martino
Computer Science – Internet of Things



INTRODUCTION

Introduction

- IoT and Robotics = IoRT
- A 6DOF + gripper robotics arm
- A smart car equipped with a camera
- A warehouse management system with 3 different sorting paths
- An architecture supporting interaction between sensing devices and robotics arm

HARDWARE

List of Components

01

Raspberry Pi

It’s a small, powerful and cheap computer board. The model used in this project is the 3B with Wi-Fi module integrated on board. Raspbian in the OS in use, and one of the great thigs about the board is that it has a wide range of usage.

List of Components

02 **1x Arduino Rev2**

Arduino UNO WiFi Rev.2 is the one-stop-solution for many of the basic IoT application scenarios.

03 **IR KY-032**

It is an infrared sensor used for the detection of obstacles, as it is equipped with an IR emitter and an IR receiver that allow Arduino to signal the presence of objects in the range of this sensor.

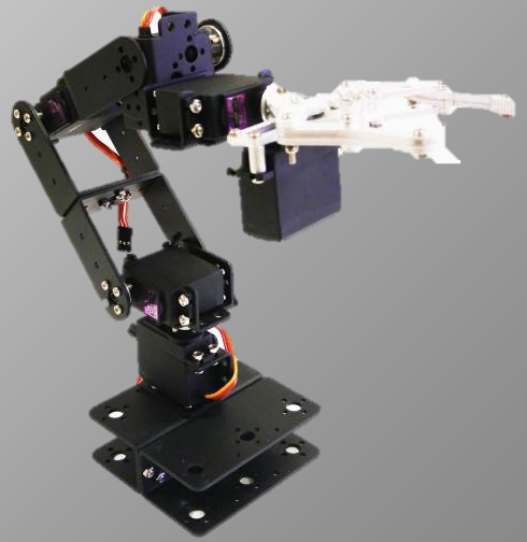
04 **DC Motor**

05 **Battery kit**

List of Components

06 **Robot chassis kit**

The robot skeleton.



07 **6x Servo MG996R**

It is a motor that can be turned to a specified position, from 0 to 180 degree. It is used in remote control (RC) world and also for robotics and automation projects.

08 **PCA9685 I2C Control**

It is a 16 PWM output channel mainly used to control LEDs, but it can be used also to control servos or as power supply at 6V.

List of Components

09 **1x Arduino Rev2**

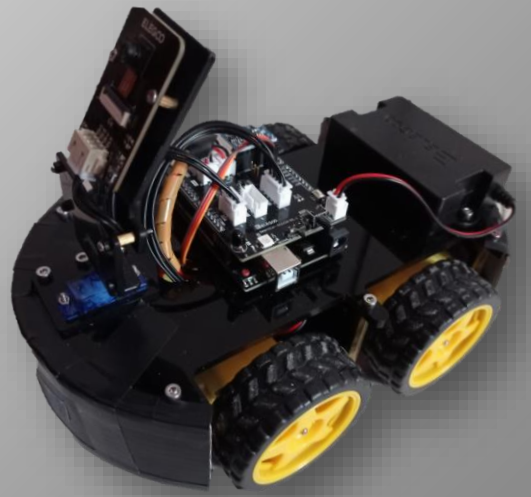
Arduino UNO WiFi Rev.2 is the one-stop-solution for many of the basic IoT application scenarios.

10 **1x ESP32 Cam**

It is an infrared sensor used for the detection of obstacles, as it is equipped with an IR emitter and an IR receiver that allow Arduino to signal the presence of objects in the range of this sensor.

11 **4x DC Motor**

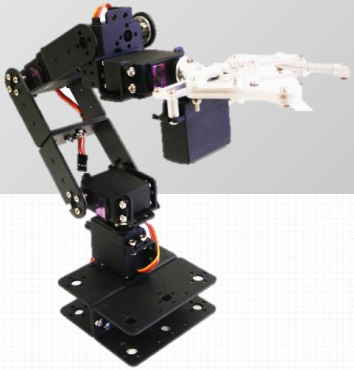
12 **Car and battery kit**

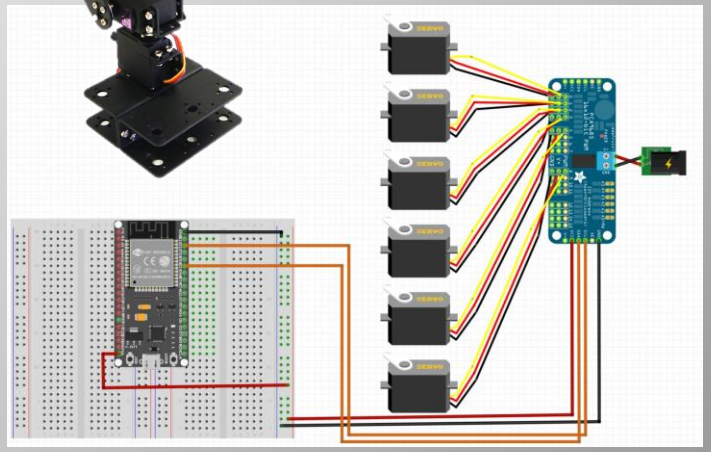


Hardware Connection

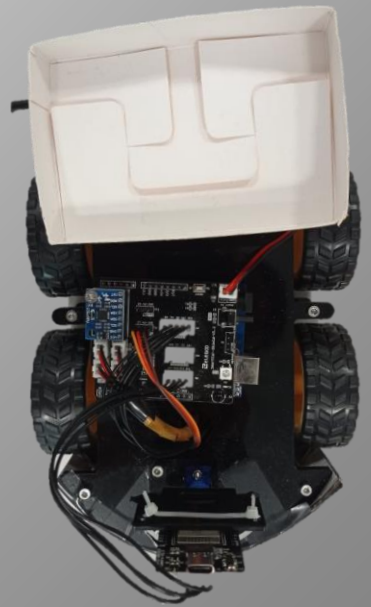
Conveyor belt

Robot Arm Node





Smart car

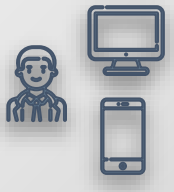


ARCHITECTURE

Architecture

Router & Internet

Operator



Dashboard

NodeRed & MQTT broker

Topics

robotics/wristPitchArm	robotics/wristRollArm
robotics/baseArm	robotics/shoulderArm
robotics/elbowArm	robotics/gripperArm
arm/controller	arm/conveyor
arm/car	car/control
car/camera	car/parameters

Subscribe
Publish

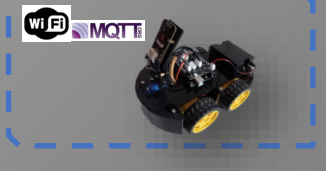
Conveyor belt

Subscribe
Publish

Robot Arm Node

Subscribe
Publish

Smart car



SOFTWARE

WiFi & MQTT Connection – Robotic Arm

```
//=========================================================================================
#include <Wire.h> // Include Wire Library for I2C Communications
#include <WiFiClient.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <Adafruit_PWMServoDriver.h> // Include Adafruit PWM Library
//=========================================================================================
#define MIN_PULSE_WIDTH       650
#define MAX_PULSE_WIDTH       2350
#define FREQUENCY             50
//=========================================================================================
const char* mqtt_server = "raspberrypi"; // Provide localhost name of the broker devices
const char* mqtt_user = ""; // Provide the username of the broker devices
const char* mqtt_pass = ""; // Provide the password of the broker devices
const int mqtt_port = 1883; // Provide the port number of the broker devices (1883, 8883)
const char* ssid = "WiFi-LabIoT";
const char* password = "sljzsjkw5b";
//-----------------------------------------------------------------------------------------
```

```
void setup_wifi()
{
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
void reconnect()
{
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("Arduino Robotic Arm")) {
      Serial.println("connected");
      // Subscribe
      client.subscribe("arm/baseArm");
      client.subscribe("arm/shoulderArm");
      client.subscribe("arm/elbowArm");
      client.subscribe("arm/wristPitchArm");
      client.subscribe("arm/wristRollArm");
      client.subscribe("arm/gripperArm");
      client.subscribe("arm/controller");
      client.subscribe("arm/conveyor");
      client.subscribe("arm/car");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void setup()
{
  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);

  // Setup PWM Controller object
  pwm.begin();
  pwm.setPWMFreq(FREQUENCY);
}
```

Movement - Robotic Arm

MQTT Callback

```
void callback(char* topic, byte* message, unsigned int length)
{
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }

  Serial.println();

  if (String(topic) == "arm/controller" && messageTemp == "Manual") {
    arm_controller = "Manual";
  }
  else if (String(topic) == "arm/controller" &&messageTemp == "Automatic") {
    arm_controller = "Automatic";
  }

  if (arm_controller == "Manual") {
    if (String(topic) == "arm/baseArm") {
      servoAngle[0] = messageTemp.toInt();
      moveMotorDeg(servoAngle[0], motorBase, delayTime);
    } else if (String(topic) == "arm/shoulderArm") {
      servoAngle[1] = messageTemp.toInt();
      moveMotorDeg(servoAngle[1], motorShoulder, delayTime);
    } else if (String(topic) == "arm/elbowArm") {
      servoAngle[2] = messageTemp.toInt();
      moveMotorDeg(servoAngle[2], motorElbow, delayTime);
    } else if (String(topic) == "arm/wristPitchArm") {
      servoAngle[3] = messageTemp.toInt();
      moveMotorDeg(servoAngle[3], motorWristPitch, delayTime);
    } else if (String(topic) == "arm/wristRollArm") {
      servoAngle[4] = messageTemp.toInt();
      moveMotorDeg(servoAngle[4], motorWristRoll, delayTime);
    } else if (String(topic) == "arm/gripperArm") {
      servoAngle[5] = messageTemp.toInt();
      moveMotorDeg(servoAngle[5], motorGripper, delayTime);
    }
  }
```

Movement

```
// Function to move motor to specific position
void moveMotorDeg(int moveDegree, int motorOut, int delayTime)
{
  int pulse_wide, pulse_width;

  // Convert to pulse width
  pulse_wide = map(moveDegree, 0, 180, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH);
  pulse_width = int(float(pulse_wide) / 1000000 * FREQUENCY * 4096);

  //Control Motor
  if (currentPos[motorOut] == pulse_width) {
    return;
  }
  else if (pulse_width > currentPos[motorOut]) {
    for (int pos = currentPos[motorOut]; pos < pulse_width; pos++) {
      pwm.setPWM(motorOut, 0, pos);
      delay(delayTime);
    }
  }
  else if (pulse_width < currentPos[motorOut]) {
    for (int pos = currentPos[motorOut]; pos > pulse_width; pos--) {
      pwm.setPWM(motorOut, 0, pos);
      delay(delayTime);
    }
  }

  currentPos[motorOut] = pulse_width;
  servoAngle[motorOut] = moveDegree;

  Serial.println("Analogue Servo Position: ");
  Serial.println(pulse_width);

}
```

Conveyor belt

MQTT callback

```
void callback(char* topic, byte* message, unsigned int length)
{
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }

  Serial.println();

  if (String(topic) == "conveyor" && messageTemp == "active") {
    Application_FunctionSet.conveyor_control("Backward", conveyor_speed);
  }
  else if (String(topic) == "conveyor" && messageTemp == "deactivate") {
    Application_FunctionSet.conveyor_control("stop_it", 0);
  }
  else if (String(topic) == "car/conveyor" && messageTemp == "pick_on") {
    pickObject = "pick_on";
  }
}
```

Object detection & MQTT communication

```
void loop()
{
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 5000) {
    lastMsg = now;
  }

  objectDetectIR = digitalRead(IR);

  if (objectDetectIR == LOW && pickObject == "pick_on") {
    Application_FunctionSet.conveyor_control("stop_it", 0);
    client.publish("conveyor/object", "green");
    client.publish("arm/conveyor", "pick");
    pickObject = "pick_off";
  }
}
```

Movement – Smart car

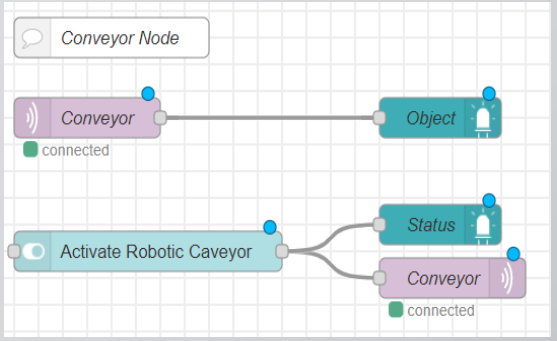
MQTT Callback

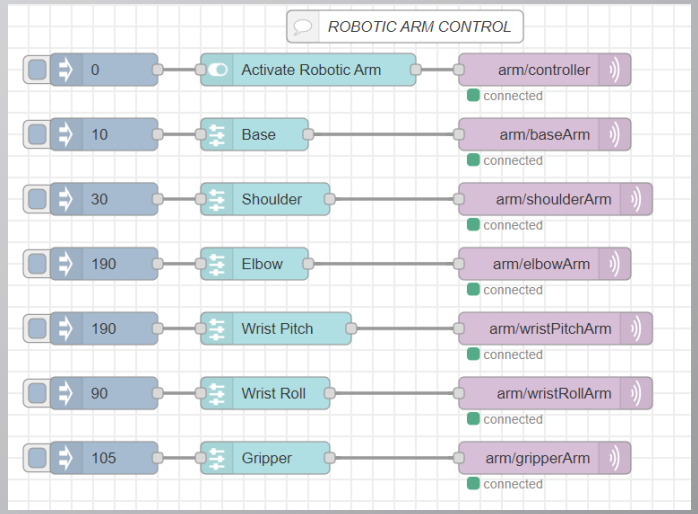
```
void callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }

  if (String(topic) == "car/control" && messageTemp == "Manual") {
    activate_robotic_car_control = "Manual";
  }
  else if (String(topic) == "car/control" && messageTemp == "Automatic") {
    activate_robotic_car_control = "Automatic";
  }
  if (String(topic) == "car/parameters" && messageTemp == "param_on") {
    activate_robotic_car_parameters = "param_on";
  }
  else if (String(topic) == "car/parameters" && messageTemp == "param_off") {
    activate_robotic_car_parameters = "param_off";
  }
  Serial.println();

  // If a message is received on the topic esp32/output, you check if the message is either "on" or "off".
  // Changes the output state according to the message
  if (activate_robotic_car_control == "Manual") {
    if (messageTemp == "Forward") {
      //Serial.println(messageTemp);
      Application_FunctionSet.car_control("Forward", car_speed);
    }
    else if (messageTemp == "Backward") {
      //Serial.println(messageTemp);
      Application_FunctionSet.car_control("Backward", car_speed);
    }
    else if (messageTemp == "Left") {
      Serial.println(messageTemp);
      Application_FunctionSet.car_control("Left", car_speed);
    }
```

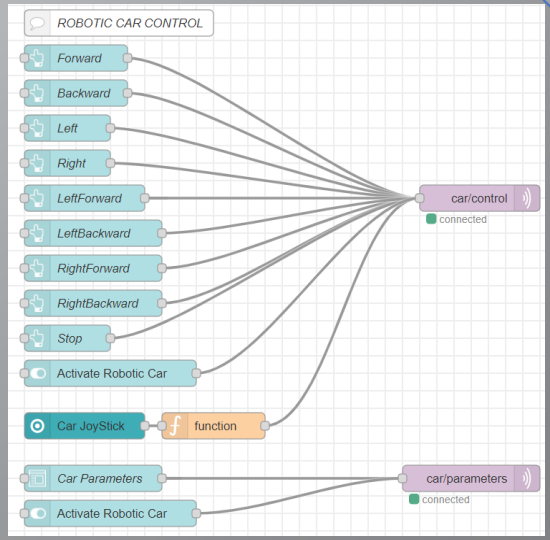
Movement

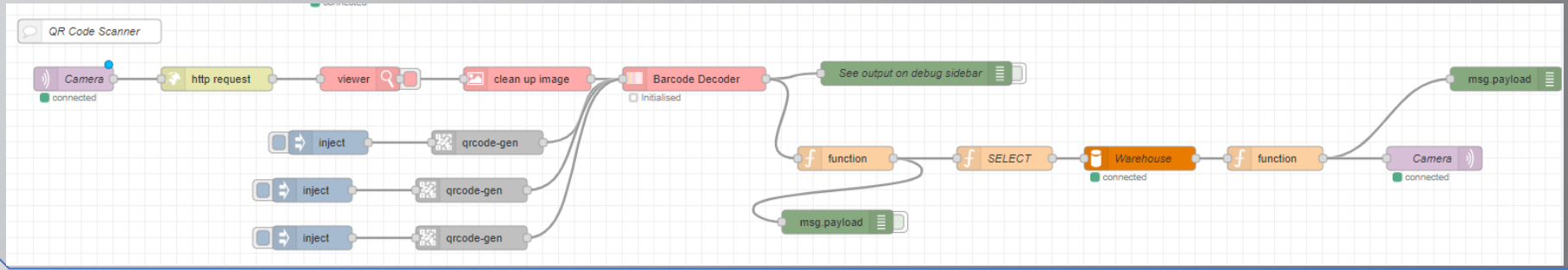
```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  //if initialization parameters arrived, the car can starts
  if (car_direction != 0) {
    carStarts = true;
  }

  if (carStarts && qr_code != "") {
    Serial.println("Start!");

    if (placeObject == "place_on") {
      client.publish("arm/car", "place");
      placeObject = "place_off";
    }

    if (objectPlaced) {
      //here we must put the code that will let the car search and then follow the line
      while (path_finder == 0) {
        path_finder = Application_FunctionSet.find_path(qr_code[0], path_finder);
      }

      while (path_finder == 1) {
        path_finder = Application_FunctionSet.ApplicationFunctionSet_Tracking();
      }

      if (path_finder == 2 && stopAfterReturnPath == false) {
        delay(5000);
        path_finder = Application_FunctionSet.moveIt('A', car_direction, car_distance, car_speed_hi);
        stopAfterReturnPath = true;
        //qr_code = '0';
      }

      else if (stopAfterReturnPath == true) {
        qr_code = "";
        path_finder = 0;
        stopAfterReturnPath = false;
        client.publish("car/conveyor", "pick_on");
        client.publish("conveyor/object", "red");
        client.publish("conveyor", "active");
        objectPlaced = false;
      }
    }
```

NodeRED

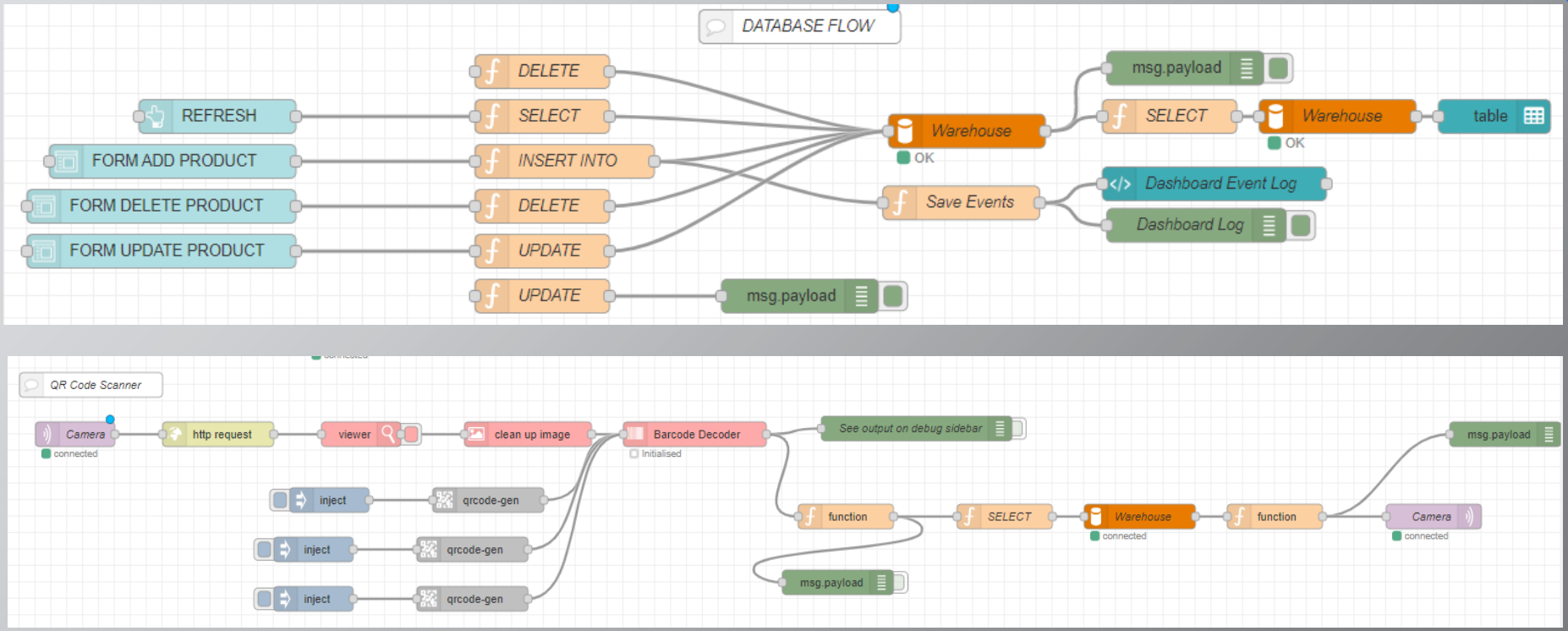

Conveyor Node
Conveyor
connected
Object
Activate Robotic Caveyor
Status
Conveyor
connected


ROBOTIC ARM CONTROL
0
Activate Robotic Arm
arm/controller
connected
10
Base
arm/baseArm
connected
30
Shoulder
arm/shoulderArm
connected
190
Elbow
arm/elbowArm
connected
190
Wrist Pitch
arm/wristPitchArm
connected
90
Wrist Roll
arm/wristRollArm
connected
105
Gripper
arm/gripperArm
connected

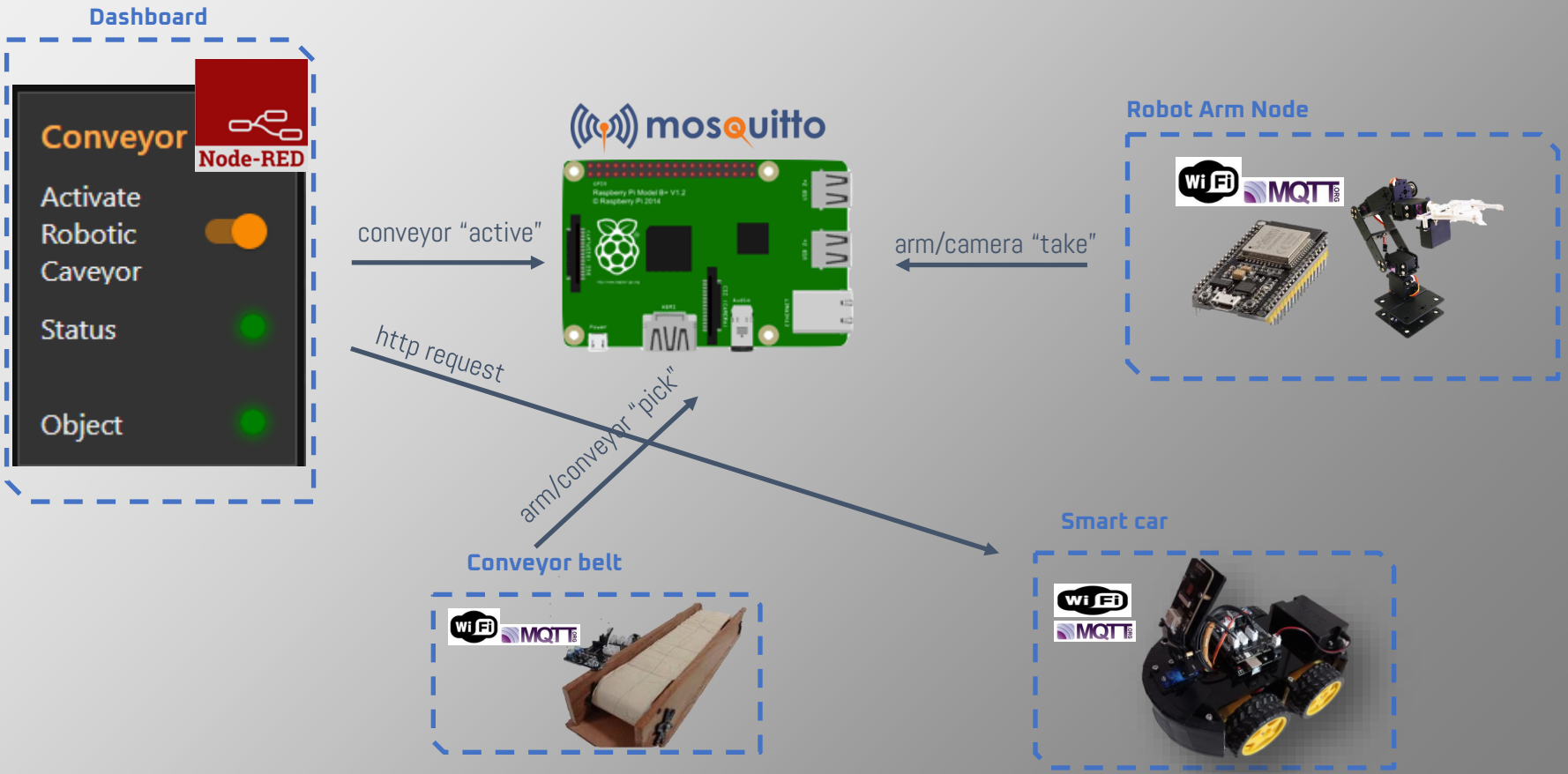

ROBOTIC CAR CONTROL
Forward
Backward
Left
Right
LeftForward
LeftBackward
RightForward
RightBackward
Stop
Activate Robotic Car
car/control
connected
Car JoyStick
function
Car Parameters
car/parameters
connected
Activate Robotic Car


QR Code Scanner
Camera
connected
http request
viewer
clean up image
Barcode Decoder
initialised
See output on debug sidebar
msg.payload
inject
qrcode-gen
inject
qrcode-gen
inject
qrcode-gen
function
SELECT
Warehouse
connected
function
Camera
connected
msg.payload

NodeRED

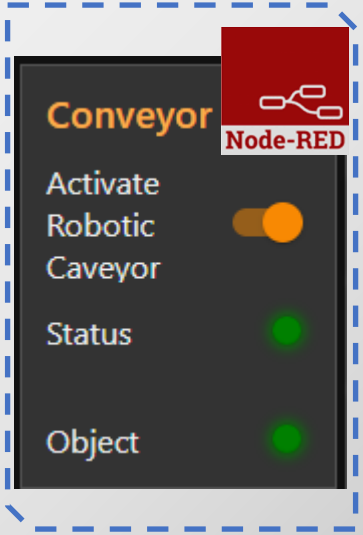

DATABASE FLOW
REFRESH
FORM ADD PRODUCT
FORM DELETE PRODUCT
FORM UPDATE PRODUCT
DELETE
SELECT
INSERT INTO
DELETE
UPDATE
UPDATE
msg.payload
Warehouse
OK
Save Events
msg.payload
SELECT
Warehouse
OK
table
Dashboard Event Log
Dashboard Log
QR Code Scanner
Camera
connected
http request
viewer
clean up image
Barcode Decoder
Initialised
See output on debug sidebar
inject
inject
inject
qrcode-gen
qrcode-gen
qrcode-gen
function
msg.payload
SELECT
Warehouse
connected
function
Camera
connected
msg.payload

Node-RED + MQTT



Node-RED + MQTT

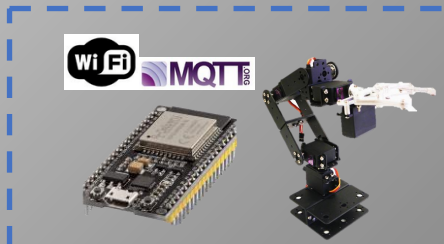
Dashboard



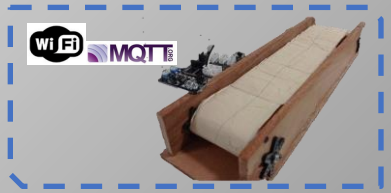
arm/car "car_on"

arm/car "place"

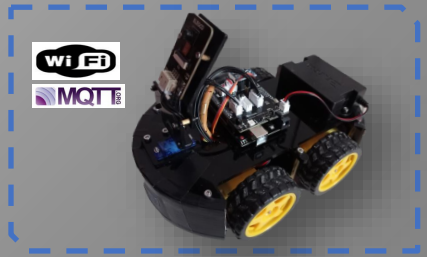
Robot Arm Node



Conveyor belt



Smart car

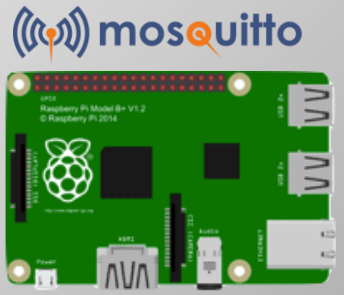


Node-RED + MQTT

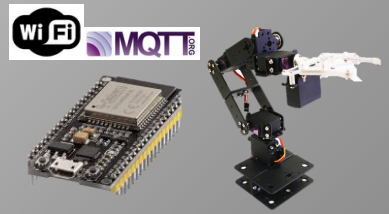
Dashboard

Database Table

Name	Number	Description	Area	ID
bottiglia	1	bottiglia d'acqua	A	7
bicchiere	2	bicchiere di carta	B	8
pupazzo	3	pupazzo orsacchiotto	C	9



Robot Arm Node



conveyor "active"

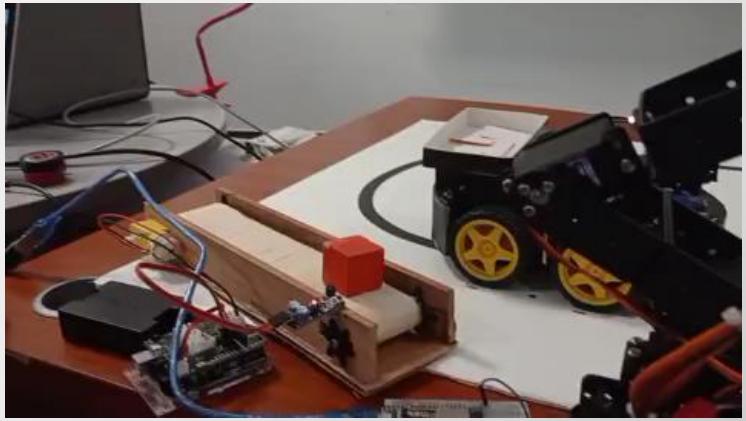
car/object "the_qr_code"

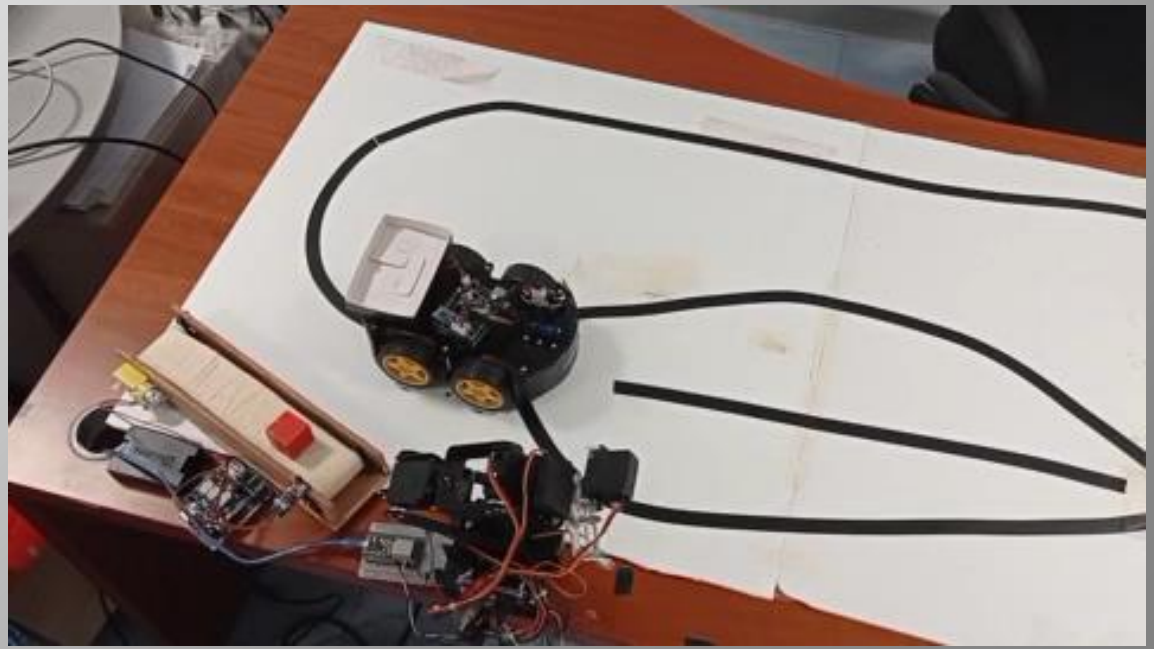
Smart car

Conveyor belt

DEMO

Demo





Conclusions

- Integrate a qr scanner to add products in the warehouse database
- Upgrade the camera with a more professional one.
- Upgrade the robotic arm with a professional one.
- Integrate the robotic arm with a camera to track the package
- Add a delivery mechanism like a drone

THANK YOU

Presented by
Gerardo Martino
Computer Science – Internet of Things