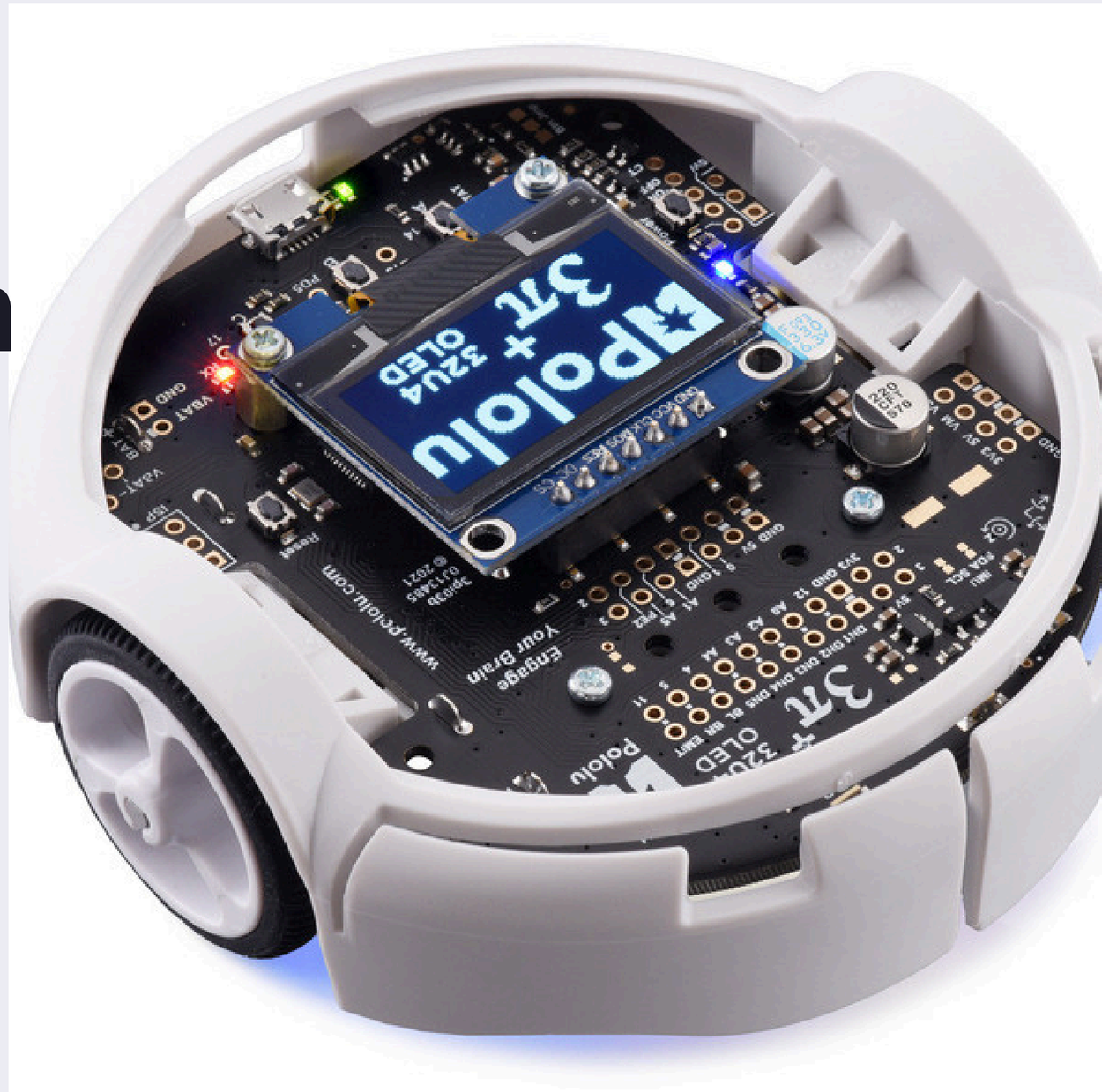
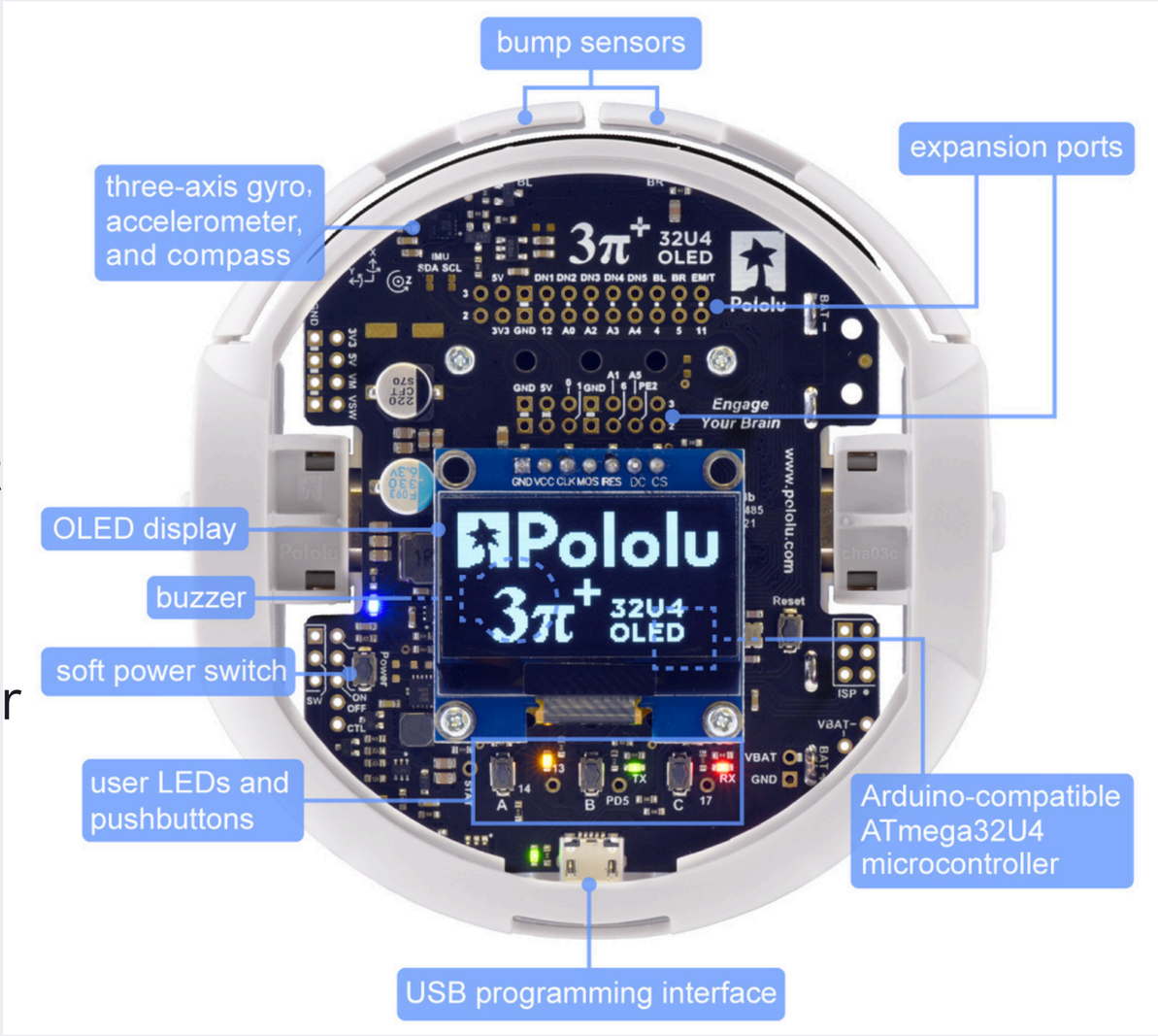
Introduction to the 3pi+ 32U4 Pololu Robot Hyper Edition

This robotic platform is an advanced version of the popular 3pi+ 32u4 robot developed by Pololu. It features enhanced sensors, processing power, and enhanced navigation capabilities, making it a powerful tool for a variety of applications. This presentation will dive into the hardware design, software implementation, and autonomous functionality of this exciting robot.



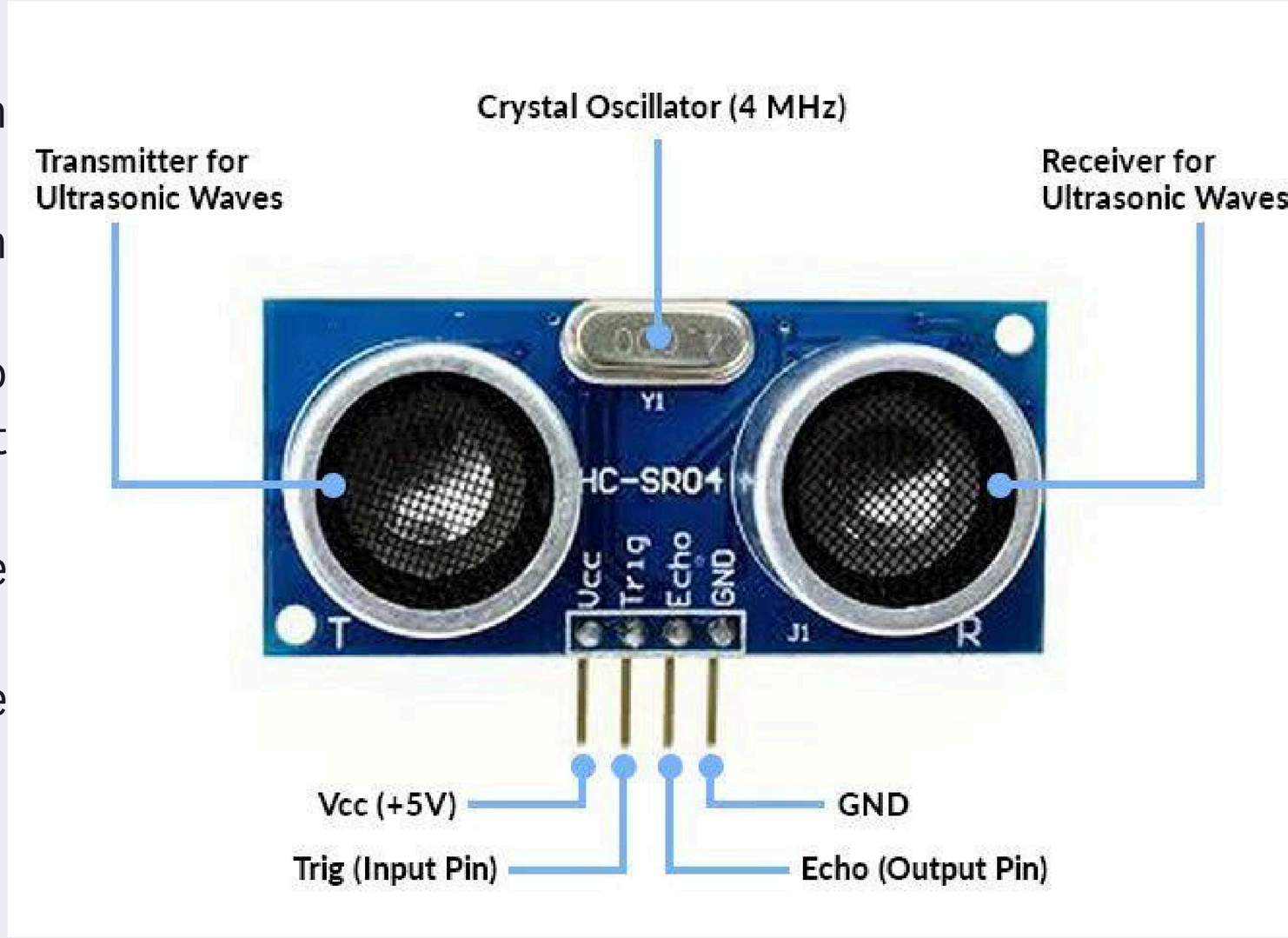
Overview of the robot

- Versatile, high performance, user programmable robot, with a diameter of 9.7cms
- Arudino compatible bootloader and ATmega32U4 AVR microcontroller chip that can be programmed using arudino
- Two H-bridge motor drives, accelerometer, gyro, magnetometer, five reflectance sensor at the bottom, bump sensors.
- Hyper edition of 32U4 robot with a top speed of 4m/s.



Overview of Ultrasonic Sensor

- Operate beyond the range of human hearing (>20 kHz)
- Uses a transducer that can both emit and receive ultrasonic waves
- HC-SR04 (5V) provides 2cm to 400cm of non-contact measurement functionality
- Echo(Receive) pin to pin-2 of the board
- Trigger(Send) Pin to pin - 3 of the board
- Connected using jumper wires



Our Polulu 3pi+ 32U4 Hyper Edition Robot

After connecting HC-SR04 ultrasonic sensor to the robot

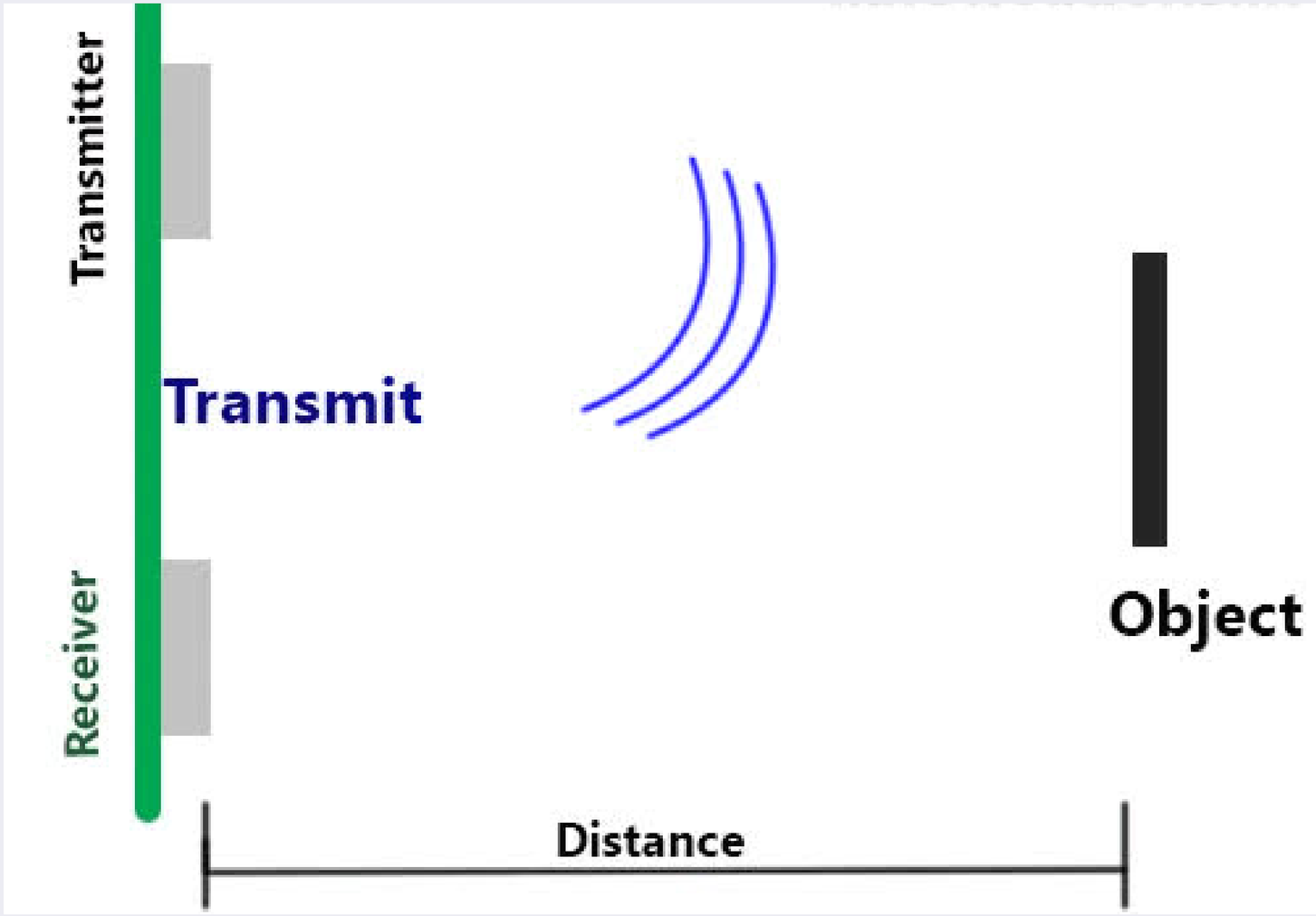
Trigger pin to digital pin3
Echo pin to digital pin2

Working and Distance Meaurement

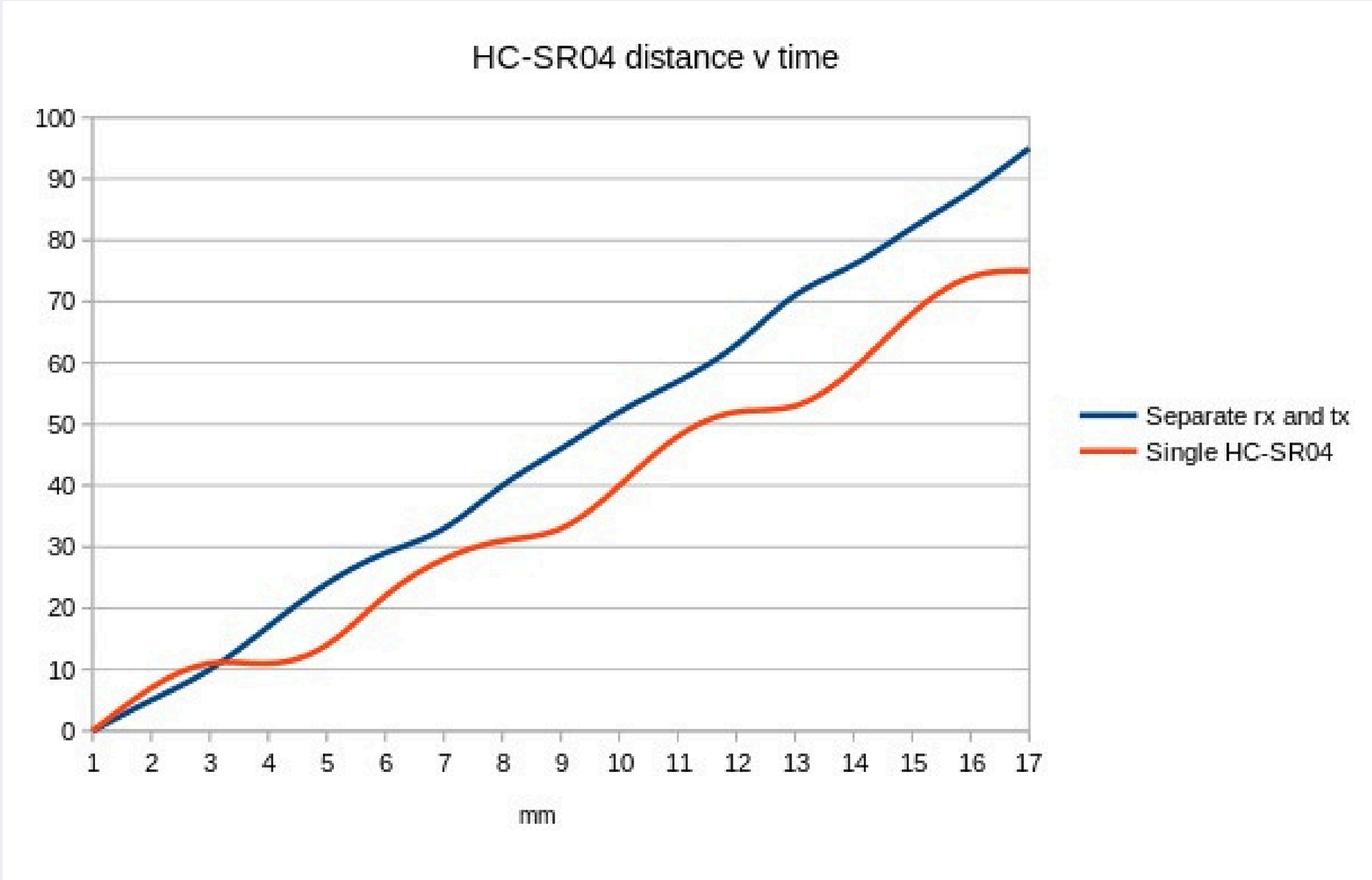
Speed of sound (air) = 0.034cm / µsec

Distance = Signal Speed * Time / 2

Time measurement in microseconds



Characteristics of Ultrasonic Sensor



Real Time Data Processing from Ultrasonic Sensor

```
    Serial.print("leftSpeed : ");
    Serial.println(leftSpeed);

    Serial.print("rightSpeed : ");
    Serial.println(rightSpeed);

    Serial.print("STOP DISTANCE : ");
    Serial.println(STOP_DISTANCE);
```

```
leftSpeed : 52.00rightSpeed : 52.00  STOP DISTANCE : 20.00Distance: 69.44 cm
leftSpeed : 52.00rightSpeed : 52.00  STOP DISTANCE : 20.00Distance: 20.27 cm
leftSpeed : 52.00rightSpeed : 52.00  STOP DISTANCE : 20.00Distance: 68.58 cm
leftSpeed : 52.00rightSpeed : 52.00  STOP DISTANCE : 20.00Distance: 69.34 cm
leftSpeed : 52.00rightSpeed : 52.00  STOP DISTANCE : 20.00Distance: 69.34 cm
leftSpeed : 52.00rightSpeed : 52.00  STOP DISTANCE : 20.00Distance: 13.94 cm
```

Autonomous Navigation Algorithms

1

Obstacle Detection

The robot utilizes its sensor suite to identify and map obstacles in its environment, allowing it to plan safe and efficient routes.

2

Object Avoidance

Post object detection, robot is coded in such a way that it avoids the object, and resets i the motor speed to stop at a threshold value.

3

Path Planning

The robot finds the most optimal angle after the object avoidance utlilising data structures to plan the path by choosing the angle with maximum object distance.

Obstacle Detection

- Ultrasonic sensors emit high-frequency sound waves.
- These waves bounce off objects in the robot's environment.
- The sensor measures the time taken for the waves to return.
- Based on this time measurement, the distance to nearby objects is determined.
- distance = speed of sound * time taken / 2;
- If the distance is less than a threshold value, it forces the robot to stop and plan its route.
- Continuous analysis of returning signals enables real-time obstacle detection.

```
#include <Pololu3piPlus32U4.h>
using namespace Pololu3piPlus32U4;

const float STOP_DISTANCE = 20;
float distance = 0;

digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW)

long duration = pulseIn(ECHO_PIN, HIGH, 30000);
distance = duration * 0.0343 / 2;

if (distance <= STOP_DISTANCE) {
      leftSpeed = -preset_left_value;
      rightSpeed = 0;
      motors.setSpeeds(-(leftSpeed-1), -rightSpeed);
      delay(500);
    }
```

Obstacle Avoidance

- The code utilises motor acceleration and motor speed factor data types to adjust the speed of tjhe motor
- It ensures smooth object avoidance and navigation.
- The robot is coded to ensure that it avoids the collision by dynamically changing the motor speed using motor acceleration and deceleration.
- The motor is decelerated at a rate that provides the robot to stop at a threshold value

```
    if (distance <= STOP_DISTANCE) {
      // Smoothly adjust motor speeds based on distance
      float speedFactor = (MAX_DISTANCE - distance) /
MAX_DISTANCE - STOP_DISTANCE;
      leftSpeed = MOTOR_BASE_SPEED * speedFactor;
      rightSpeed = MOTOR_BASE_SPEED * speedFactor;
    }
    // Make sure the speeds are within the acceptable range
    if (leftSpeed < MOTOR_MIN_SPEED) leftSpeed = MOTOR_MIN_SPEED
    if (rightSpeed < MOTOR_MIN_SPEED) rightSpeed =
MOTOR_MIN_SPEED;
    // Apply motor compensation
    if (leftSpeed <= L_MOTOR_FACTOR_THRESHOLD) {
      leftSpeed *= L_MOTOR_FACTOR;
    }

    if (rightSpeed <= R_MOTOR_FACTOR_THRESHOLD) {
      rightSpeed *= R_MOTOR_FACTOR;
    }

    // Set the motor speeds
    motors.setSpeeds(leftSpeed, rightSpeed);
```

Path Planning

- After obstacle detection, the robot is coded to find the best path in its vicinity.
- It utilises data structures such as arrays, pairs, vectors, etc. to store angle alongwith the object distance at that angle.
- The algorithm finds these values for both its left and right hand side.
- After storing the values in a map, the algorithm finds the angle value that has object at maximum distance.
- If there is a conflict between some or more angle, it opts for the first angle from the left side that fits the criteria.

```
  time = millis();
  float angleR[100];
  float distR[100];
  float angleBaseR = 3.33;
  motors.setSpeeds(-40, 35);
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH, 30000);
  distance2 = duration * 0.0343 / 2;

  while(millis() - time <= 2000){
  angleR[i] = angleBaseR;
  angleBaseR += 3.33;
  distR[i] = distance1;
  i++;

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(10);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH, 30000);
  distance2 = duration * 0.0343 / 2;
}
```

Use of Bump Sensors

- There are two bump sensors present on the either side of the robot.
- The left and right bump sensors are labeled as BUMPL and BUMPR and as BL and BR in the control board.
- Used to detect collision, if collision is found then algorithm runs to change the direction of the robot

```
bumpSensors.read();
bool leftPressed = bumpSensors.leftIsPressed();
bool rightPressed = bumpSensors.rightIsPressed();

if (leftPressed || rightPressed) {
    buzzer.play(F("c32")); // Play the "c32" melody
    plan();
    float leftSpeed = -preset_left_value;
    float rightSpeed = 0;
    motors.setSpeeds(leftSpeed, rightSpeed);
}
```

Implementation Challenges

- Ultrasonic sensor detetcts objects in straight line path only.
- It may be possible that the reflected ray does not get received by the echo port due to slanting objects.
- Also possible that object can be thin than the separation between echo and trigger port separation, so it may not be detected by the sensor.
- This may cause the robot to bump to the object, thereby triggering the bump sensors to work.
- Coordination may sometimes not be achieved between the motors and the arudino board.
- Sometimes due to rough surface, the robot motors may not work properly and therefore can give different results.

Applications of Autonomous Robots

Industrial Automation

- Autonomous robots can navigate factory floors, warehouses, and assembly lines to ensure proper functioning and detecting any issue.
- Enhancing efficiency and productivity while minimizing the risk of collisions with equipment or workers.
- Can be utilised to carry industry equipments and goods to relocate to specific place.

Applications of Autonomous Robots

Surveillance and Security

- Application as Autonomous Security Robots (ASRs)
- Can be used in military for surveillance or performing tasks that are too dangerous for human beings
- Can be used as robot security cops to navigate places.

Applications of Autonomous Robots

Delivery Robots

- Autonomous delivery robots represent the future of delivery and logistics, offering unparalleled efficiency and convenience.
- These robots can navigate city streets seamlessly, mitigating traffic congestion and reducing carbon emissions making it a more greener path for delivery.
- Needs improvement in terms of handling different terrains and obstacles and anti theft mechanism
- Many companies like Starship Technologies, are working to enhance the future of delivery by utilizing its automation

Challenges and Troubleshooting

Sensor Calibration

Ensuring accurate and consistent sensor readings to enable reliable obstacle detection and environmental mapping.

Power Management

Optimizing the robot's power consumption to maximize runtime and ensure stable operation during autonomous navigation.

Software Optimization

Refining the Arduino code to improve decision-making algorithms, data processing, and overall system performance.

Testing and Validation

Thorough testing in various environments to identify and address any issues or edge cases that may arise during autonomous operation.

Conclusion and Future Enhancements

Capabilities Achieved	Object detection and avoidance	Smart route planning	Increased Processing Power
Future Enhancements	Integrating machine learning	Installation of ESP-6 WiFi modules for remote instructions	Implementing 2D mapping for optimal path planning

In conclusion, the 3pi+ 32u4 Pololu Robot Hyper Edition has demonstrated impressive autonomous navigation capabilities, leveraging advanced sensors and a powerful microcontroller. By continually improving the software, expanding the sensor suite, and incorporating machine learning, this robotic platform holds great promise for future advancements in autonomous systems.

Arduino Coding and Functionality

Sensor Data Acquisition

1

The Arduino code interfaces with the various sensors, continuously gathering real-time data about the robot's surroundings.

Motion Control

2

The code implements advanced algorithms to translate sensor inputs into precise motor control, enabling smooth and responsive movements.

Decision-Making and Navigation

3

The software integrates complex decision-making processes, allowing the robot to plan and execute autonomous navigation strategies.


Thank
You