Sensors on wheels

Mikhail Deriabin, Leo Hannolainen

Abstract

The first problem which was tried to solve is to speed-up a process of measuring different quantities in multiple points of an area. The second problem is the collected data visualization. All types of businesses are facing the same problem – inefficiency of processes, which they are always trying to solve. The quantities measuring is one of such processes and our solution helps to save time and other recourses of actual measuring. Our solution is to provide a driving robot with different sensors attached, as well as a software for sending sensor values to the server, saving them, and generating a map displaying measuring results in different points in the area. As a result, in about 7-10 minutes user will get the map for area of 1.5x3.0 square meters.

1 Introduction

The first problem which was tried to solve is to speed-up a process of measuring different quantities in multiple points of an area. The second problem is the collected data visualization. All types of businesses are facing the same problem – inefficiency of processes, which they are always trying to solve. The quantities measuring is one of such processes and our solution helps to save time and other recourses of actual measuring. Example of such business case may be building companies, which need to verify the floor evenness, when the apartments are ready. Another example is an inspection of the noise level and air quality in different places of a workspaces such as offices. Our solution is to provide a driving robot with different sensors attached, as well as a software for sending sensor values to the server, saving them, and generating a map displaying measuring results in different points in the area. Maps are generated basing on an ultrasonic distance sensor measurements. These maps can be saved for future uses. As a result, in about 7-10 minutes user will get the map for area of 1.5x3.0 square meters. The same operation may be done manually using for example Arduino board connected to a laptop and in addition there is a need to draw the map itself and somehow store the data, but it will take much more time.

2 Goals

2.1 Plan and assemble a robot

Plan the robot architecture, its electric and mechanical systems and build the robot.

2.2 Create software architecture for the robot

Create a software architecture for the robot and components such as sensors for easier maintaining and scaling the project in the future.

2.3 Robot connection to the server

Implement robot connection to the server using the MQTT protocol for receiving commands from the application and sending sensors data.

2.4 Create communication messages format

Create a format of messages in which they will be sent between robot and the application.

2.5 Create an application

Create an application with robot control panel and map pages for interacting with the robot.

2.6 Create map generating logic

Create a logic with which maps will be generated based on the ultrasonic distance sensor data received from the robot

2.7 Save generated maps

Provide a database architecture for saving maps and implement it.

3 System Architecture and Design

3.1 Hardware

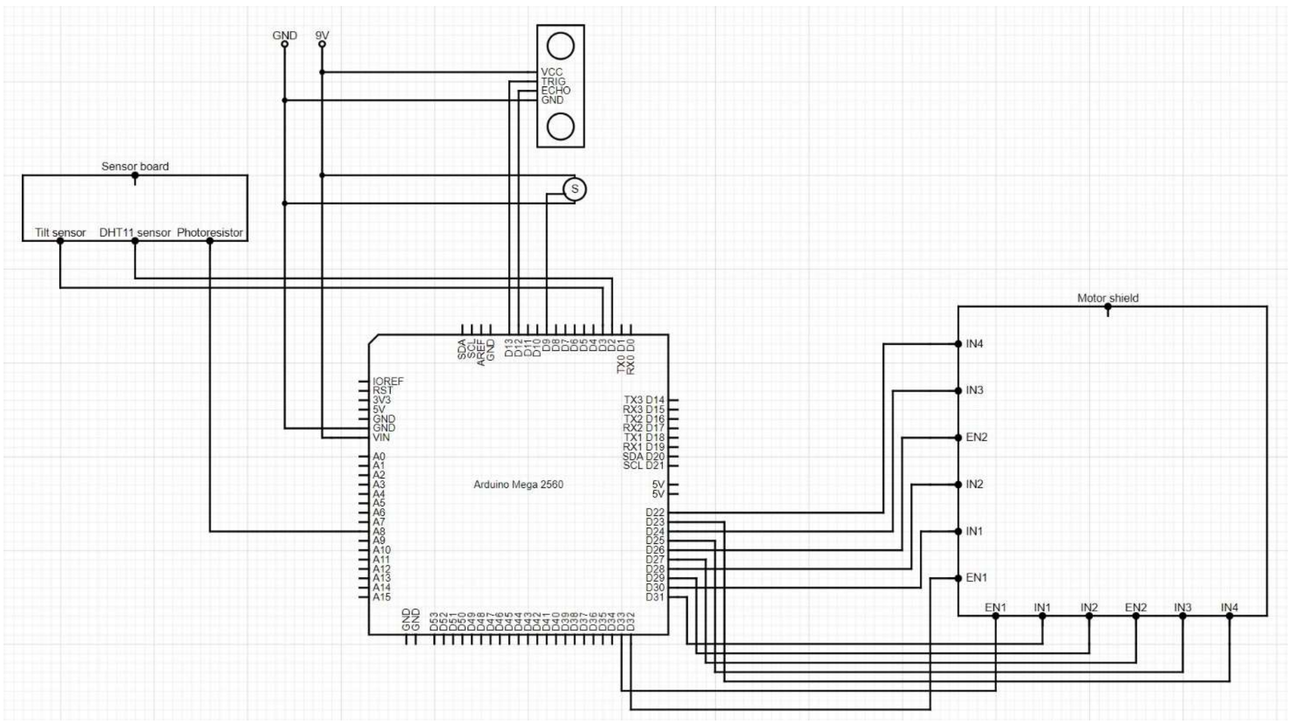
Component/Device	**Use**
Arduino Mega Wi-Fi R3	All robot systems control. Communication with server
4x DC Motors	Robot driving
Servo SG90 Motor	Ultrasonic sensor rotation
2x L293D chips	Connecting DC motors to the Arduino board
18V Einhell accumulator	Powering the robot
2x DC to DC converters	Converting accumulator 18V to 9V and 5V for the robot
Ultrasonic distance sensor HC-SR04	Measuring distance between robot and walls
Mercury tilt sensor	Determining is robot tilted or not
Photoresistor	Measuring light intensity
Temperature and humidity sensor DHT11	Measuring temperature and humidity
Raspberry Pi 4	MQTT broker and API server

As a MQTT broker we are using Raspberry Pi.

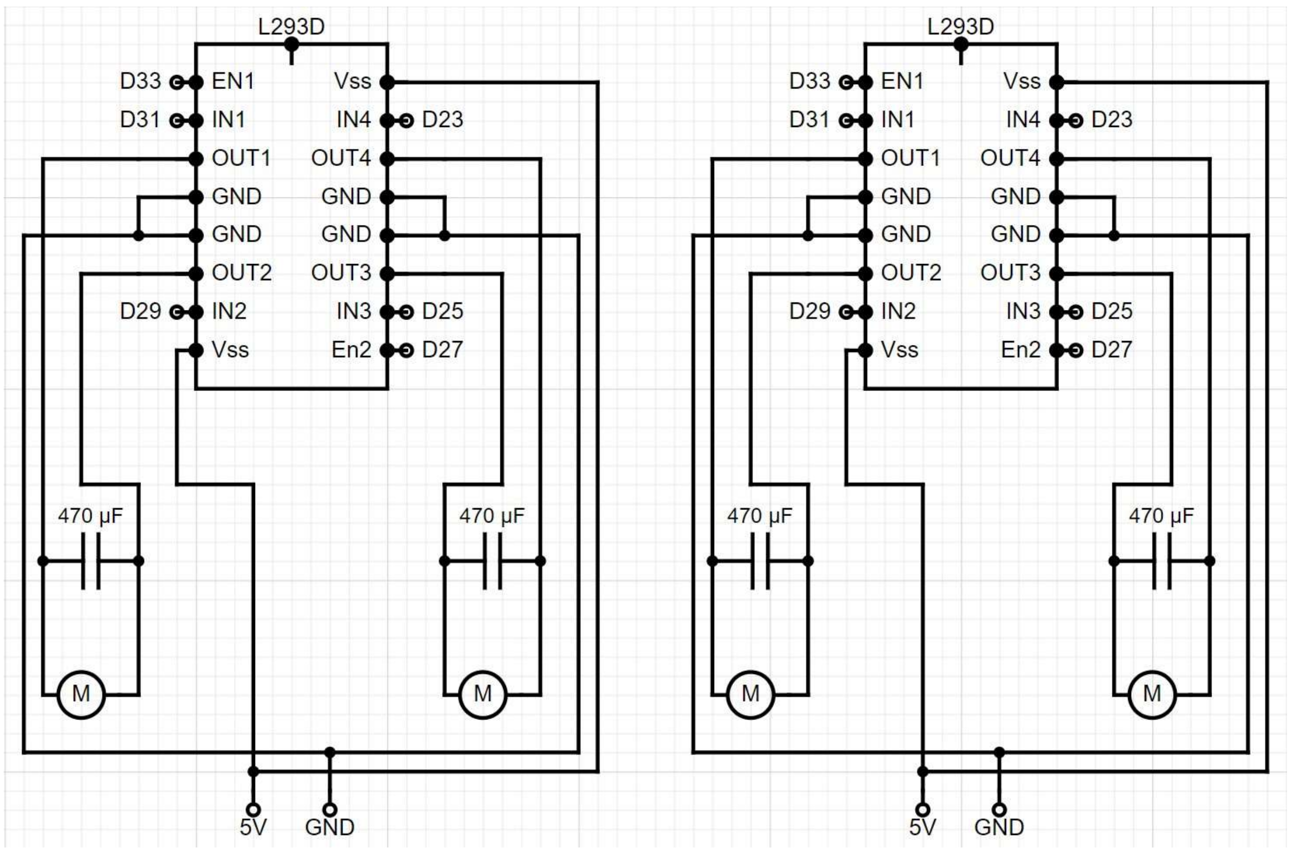
All the components of robot are connected together with wires and pin connectors. Some components are soldered together on a soldering board. In total there is 4 parts of the robot system:

1. Arduino board
2. Motor shield board (2x L293D chips + capacitors)
3. Power board (with backward current protection)
4. Sensor board (Tilt, photoresistor and DHT11 sensors)

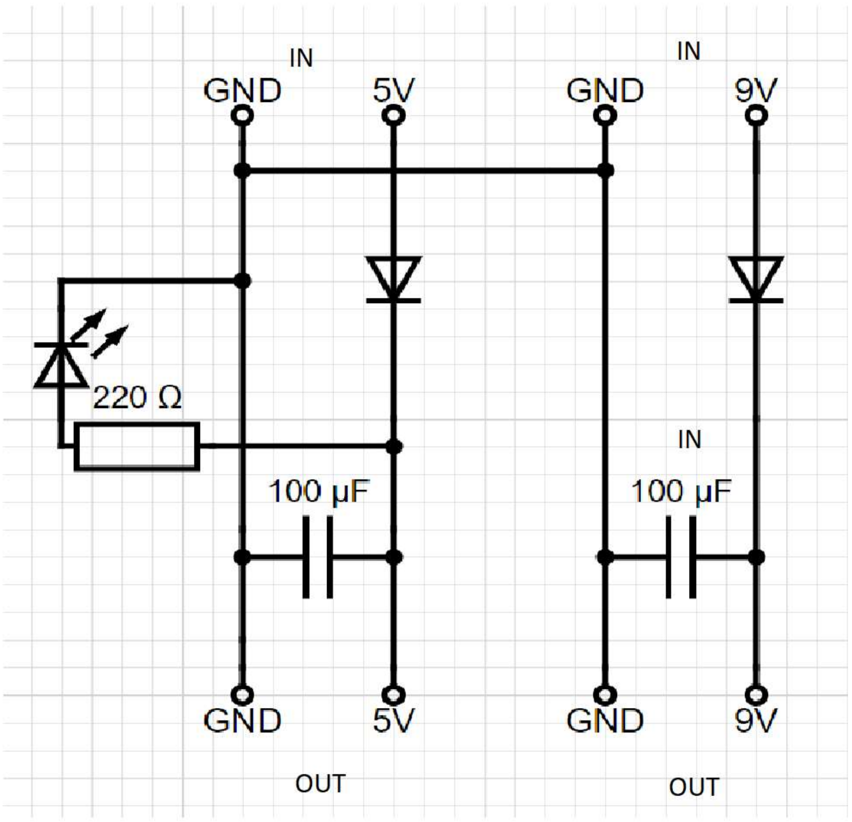
Below is a schematic of the robot boards and they connections.



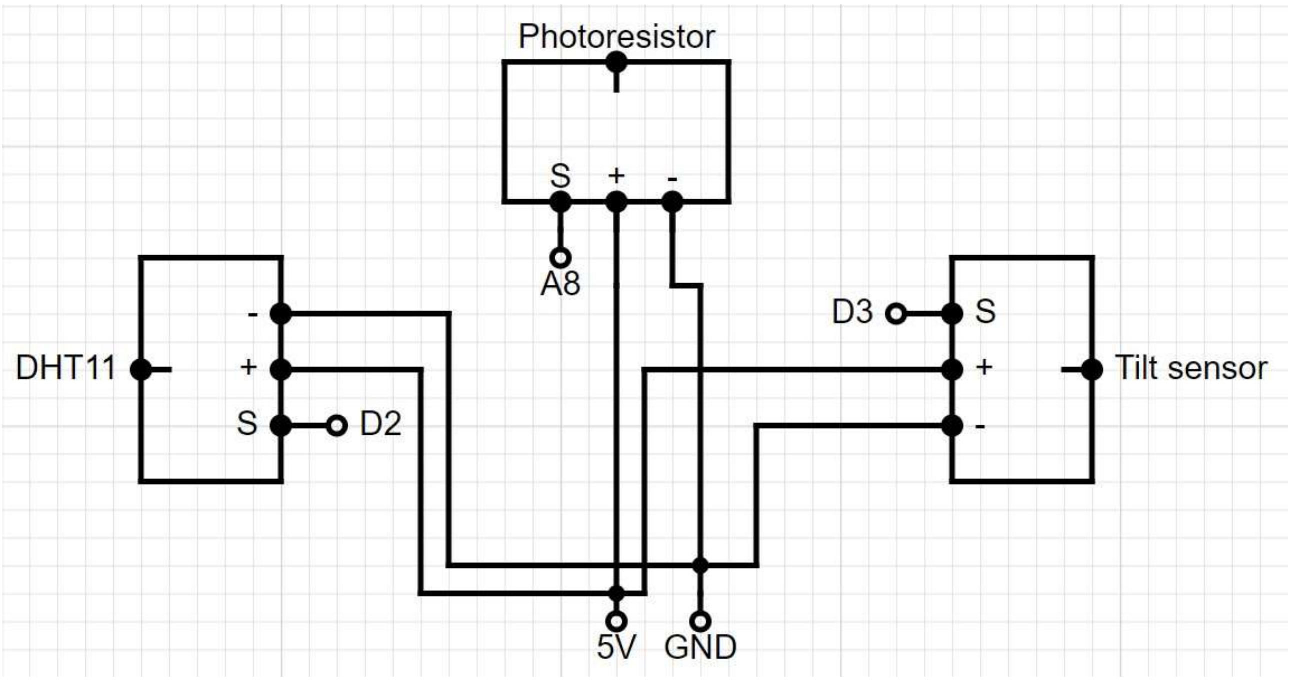
Arduino board connections



Motor shield board



Power board



Sensor board

3.2 Software

Library/framework	Use
Arduino libraries	Programming the robot and sensors
ESP Library	Communication with a server
Mosquitto MQTT	MQTT broker
Express + MariaDB	API (manipulating with maps)
Electron + React	Application

Since we are using Arduino with embedded ESP board, we are using ESP board for communication with a MQTT broker and sending forward the messages to the Arduino via Serial. As an MQTT broker we are using Mosquitto MQTT.

Application is built with Electron and React. We save generated maps and manipulating with its data by an API made on Express and MariaDB.

4 Addressing Challenges

4.1 Building robot

In the process of building robot many small challenges occurred. For example, ultrasonic sensor placing on a servo motor so, that the wires soldering points on the sensor will not brake while servo motor is moving. The solution was to add wires draft shield and to make them long enough to prevent wires tension. Many such different problems were solved in a many different ways.

4.2 Communication between Arduino and ESP boards

One of the biggest problem to solve was to make Arduino and ESP work together.

The first part of the problem was to update the ESP firmware, so that it could work with some relatively newer libraries. It was solved by using ESP own program for this purpose and tutorials.

The second part was communication itself. We were decided to use ESP board as a communicator with outside world since it has a Wi-Fi antenna. So, we created a little communication protocol, which describes the form of the messages between Arduino and ESP.

4.3 Robot moving

Another problem was to get robot move straight forward and turning on exactly 90 degrees. Get exactly 90 degrees turn was tried to solve with timers by measuring how long the robot is turning, but this method did not work. The only solution at the moment for these problems is to give user opportunity to align robot by adding couple buttons for a little turning it to the left or right.

4.4 Creating multiple MQTT clients

The application created multiple MQTT clients. The solution was to use Singleton pattern for creating only one MQTT client.

5 Performance Evaluation and Testing Results

As a relative point for the project performance evaluation the next situation was chosen. User have an Arduino device with some sensors connected to it. The Arduino device is connected to the laptop, where this device is sending data from sensors. After receiving sensors data user save it and move to another point to measure values again. After measuring process, user draw a map and points with sensors data, where the user measured them. Time consumption of this project is about 30 min for 1x3 m area with 10 different points of measurement.

The same operation was performed but using the robot and application. The result is about 7-8 min for the same area and points count.

The performance of the system for the operation is about 3 times faster with robot and application, than doing it manually.

6 Concluding Remarks and Avenues for Future Work

As a result of the project, we got working system, which outperform manual measuring process significantly.

The project itself is still raw and will be improved in the next versions. The next features to implement is improving robot movement accuracy by adding helping sensors such as compass and motor speed sensors. To robot may be added auto measuring mode for measuring everything automatically without need to control it by person. Application functionality may be increased by adding for example user profiles.

The companies may start using this project by creating the robot, adding sensors to it, write a script and download the application. Script and the application can be found in the GitHub repository as well as more detailed description and instructions.

7 Availability

Video: https://www.youtube.com/watch?v=5dwgvSX0VvQ

GitHub: https://github.com/MikhailDeriabin/RoboCar