# Team Five Final Report

Ziyu Wang Zijian Wang Shenjie Bao

## Team 5

Team Uppsala: Tim Svensson, Sebastian Persson, Sudarsan Bhargavan

Team Turku: Ziyu Wang, Zijian Wang, Shenjie Bao

Team Hanoi: Tran Thanh Cong, Trần Việt Hải, Tuấn Dũng Ngô

Team leader**:** Tim Svensson

Subleader: Tim Svensson, Tran Thanh Cong, Ziyu Wang

## Tools used:

**Runestone**: Project site, time log, wiki

**Github**: Code repository

**Slack**: Communication, meetings

**Google Hangout**: communication, meetings

**ev3dev**: Debian based GNU/Linux OS for EV3

**Arduino** IDE: Developer software for uploading Sketches

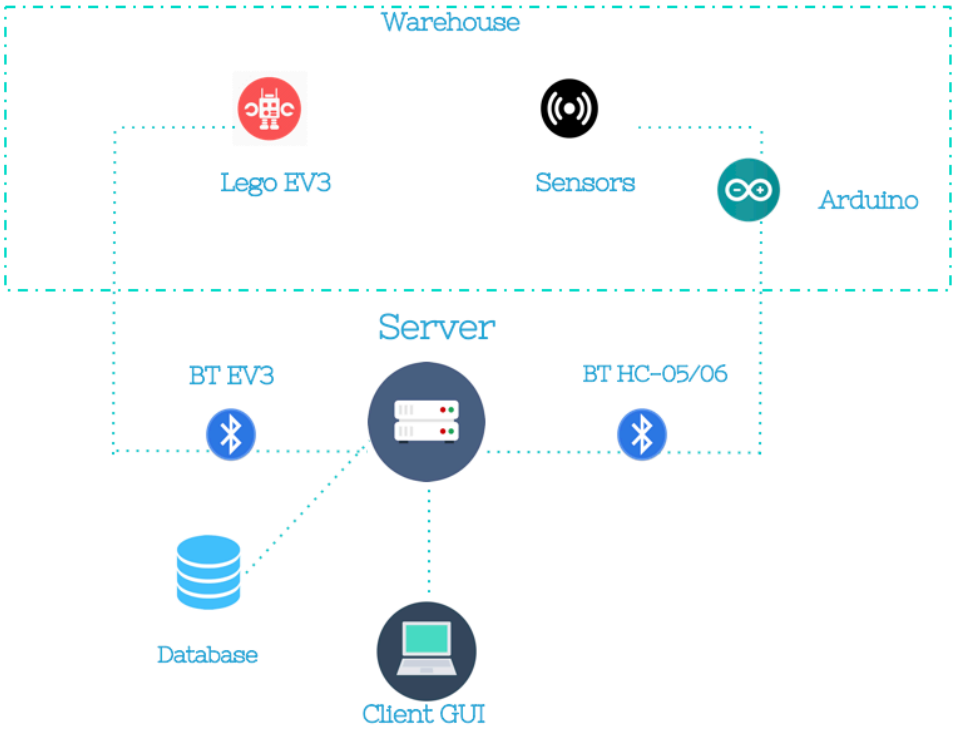
**Google Drive/Google Docs**: Documents, presentation

## 1. Introduction

In the course of Advanced Sensor Network, which is also called Global Project Management by Upsala and Hanoi, is a course that requires teamwork through around world students to finish a project.

And in our discussion, each team should have a division of labor, the Uppsala University focus on WMS(Client/Server/Database). The University of Turku in Arduino and HUST is on Lego EV3.

**Overall view:**




## 2. Problem Definition

Design a distributed system which can perform the smart unloading of robot and check the unloading route over the Internet.

Robot navigates its route autonomously by taking the required information from the Arduino board.

Server communicates with one or more robots to provide real-time route information and command relay between a selected robot and a remote Control Client.

## 3. System Design

In this project we will create a prototype of an automatic warehouse. The warehouse project will consist of three major parts; A sensor module, a robot and a server unit.

### 3.1 Automated Warehouse vs Manual Warehouse

So first, what is our definition of a warehouse. Our definition in this case is that it is a commercial building with space for storage of goods. The spaces are often big and plain without any height differences, so any types of goods may be placed arbitrarily. The warehouses may be used by manufacturers, importers, exporters, wholesalers etc.

The employees in a manual warehouse needs to move to pick up place, pick up the goods, and then move it to another place in the warehouse or to the "delivery dock" where it is placed before shipping. When something from the warehouse is ordered or bought, a work order is created for the workers. The workorder often includes what type of object it is, where the object has its place in the warehouse and where the destination is. If it is a reshuffle of goods or if it will be delivered to a customer, the work order includes these types of standard attributes.

An automated warehouse has the same type of purpose as the manual one, but it is controlled with a variety of computer-systems to placing and retrieving loads from a storage space in the warehouse. The automated warehouses are often applicated where there are many goods transferred in an out from the warehouse, there can be tight spaces or goods are being placed with high altitude.

The warehouse system has continuously grown through history and they have moved from being simple storehouses where they kept food or animals to multi-million-dollar facilities. Today some of the warehouses are fully automated without any manual handling. The automated warehouse is fully controlled, and storage is tracked by the company WMS (warehouse management system).

Some warehouses have a combinational automated and manual handling of the goods, e.g. if goods are delicate/fragile, or require specific skills to handle.

## 3.2 Purpose

As mentioned in the first sentence we want to create a prototype of an automated warehouse. Our sensors will collect data from the demonstrational warehouse and give feedback to our server. The server act as our WMS (warehouse management system). We have a micro robot which will illustrate the handling of the goods and the movement within the demonstrational warehouse. The robot will be fully

controlled by the WMS software. The WMS software have a basic GUI (graphical user interface) with options for the system.

The reason we want to present this small-scale solution is to show how easy an implementation of an automated warehouse can be for a company. One of the biggest interest from the buyers when investing in an automated storage and retrieval system is ROI (return of investment). How long will the payback time be? Will the stakeholders make any profits of the investment? Small companies may not even need a big automated system, a smaller version will work just as good for them.

A good thing with our system is that it can be implemented in many storage environment as the robot uses sensors to move between destinations.

## 3.3 Sensor

More specific for the sensor part of the prototype we are using a microcontroller board called Arduino which will collect the data from the sensors and send it to our server. This types of one chip computer is great for electrical projects as they are using an open-source platform. Arduino also offers a simple software for programming the device, which is called IDE (Integrated Development Environment). The programming language used for the Arduino is merely a set of C/C++ functions that can be called from our code. All sensor manufacturer provides specific libraries that can be loaded into your IDE, so their sensors works properly with your code.

The sensors in this prototype will be a simple Temperature/Humidity Sensor and a UV Sensor (only available in the Swedish version). The specific sensors are used for demonstration purpose to give the viewers a sense of what the system purpose is and how it can be extended with other types of sensors in the future.
The communication from the Arduino will be wireless with the Bluetooth technology. The Bluetooth technology is widely established around the world and most devices have support for it. The Bluetooth module has a low-power consumption and a sufficient throughput when it comes to transferring the

collected sensor data. We wanted to have the sensor module completely without wires except from the AC adapter. The wireless solution is optimal as we don't need to have a physical attachment between the server and the sensor module. The features of the sensors should be that they can be placed all around the warehouse without any restrictions or dependency of the other parts of the system.

## 3.4 Robot (Lego EV3)

Our robot is built from a Lego Mindstorms EV3 set. The set contains a brick which serves as the control center and power station for the robot, this is the brain and the heart of the robot. The set also contain 2 large motors and one medium. The large motors are used to make the robot move in all directions. The medium one is a weaker but also smaller in size and gives us more precision. This motor will be used as our pickup arm, when picking up goods in the warehouse.

The set also includes several sensors; a gyro sensor, a touch sensor, color sensor and an IR-sensor. We are going to use the color sensor and gyro sensor to help the robot with the movement.

All these parts come delivered with a big pack of matching Lego bricks. This can be used to build a robot design of your own or following the instructions to build a pre-drawn one that Lego created for you.

The EV3 have an ARM processor, on-board program storage, mini SDHC reader, USB connection, Bluetooth dongle, 4 output ports for the execution of the commands, a speaker etc.

The EV3 brick has its own operating system, which is created by Lego. Together with the Lego programming application you can create simple instructions to the robot and how the robot will move/act. We have decided to use another operating system which is called "ev3dev". The ev3dev OS is a Debian Linux based OS that run on top of the regular Lego OS. The ev3dev is flashed on a micro SD card which is inserted into the brick. When the brick boots, it will automatically load the ev3dev OS. The operating system gives us much more customizability with the programming. The ev3dev OS supports the programming language Python 3.x which we will be using to give the robot its commands from the WMS (server). We are controlling the robot wirelessly with Bluetooth.

## 3.5 Warehouse management system (Backend/Frontend)

Our WMS is the control unit of the system. It presents the data collected by the sensors and it also sends command with instructions to the robot. Our Warehouse management system (WMS) consists of a server, database and a client. The server and the client are coded with Python 3.x and will and the both parts will be acting as backend/frontend. In this project we decided to use the programming language Python. Python is easy to use but can also be very powerful. Python is a a great programming language for beginners and advanced users which suits this project perfectly as every team member comes from different programming backgrounds.

The backend will also include a database which is created with Mariadb. The GUI is a terminal based user interface, it gives the user possibility to choose between different options for the system e.g. presenting sensor data, moving the robot or present where the current location of the robot is.

## 3.6 The Overall Design

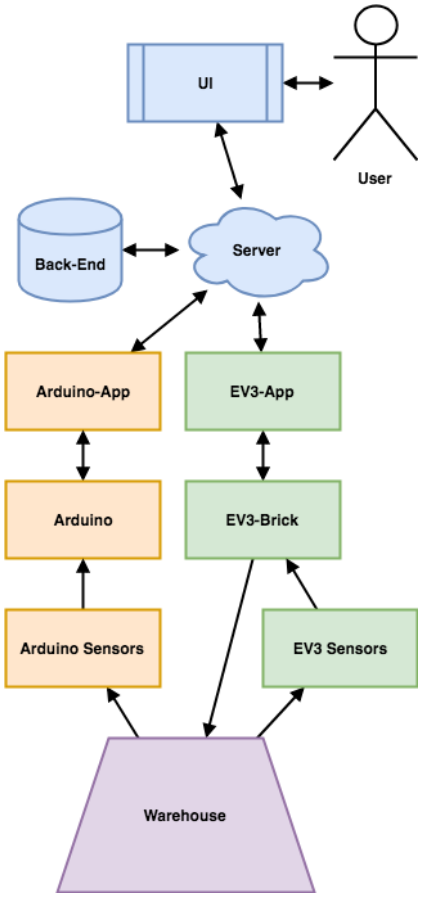



Figure 1: A graphical representation of the overall design of the system,

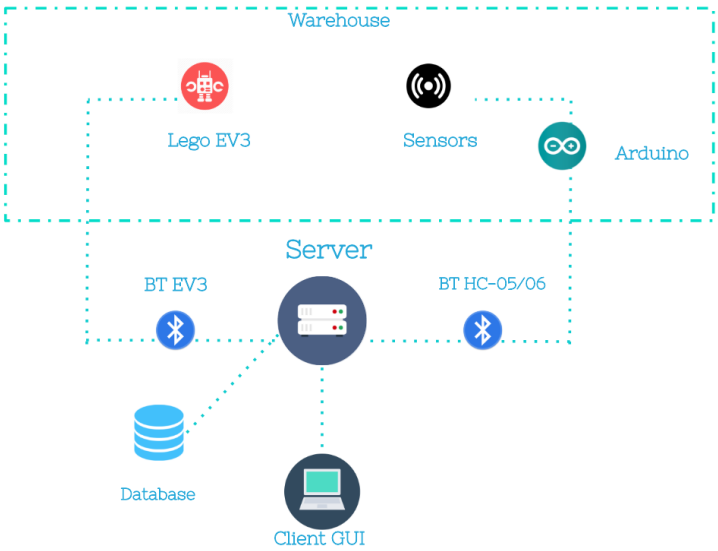


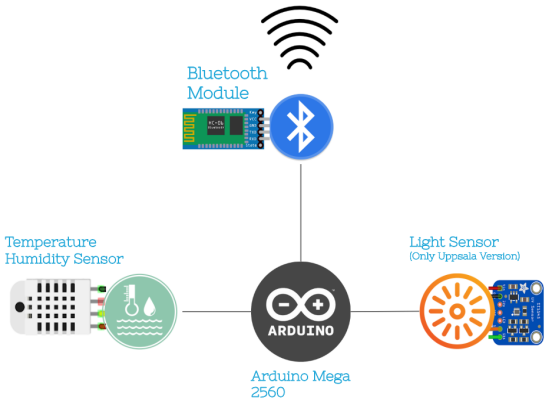

Figure 2. Alternative chart of the system design

The system is going to be divided up into 3 major parts: the Arduino, the EV3 and the Warehouse Management System (Backend/Frontend/Database). A more detailed summary for each part will follow.

## 3.7 Arduino

The arduino collects the data fetched by the Arduino sensor DHT22(AM2302) and SI1145 (Only available in Swedish version). The DHT22 measures the humidity and temperature of the environment and sends data back to the Arduino. The SI1145 is a UV/IR-light sensor

We have also connected a bluetooth device called HC-05/06 to the arduino. This makes the arduino able to send data back to a computer via bluetooth. We will program the bluetooth module in order to push the data of the sensors.




The Mega is suitable for big projects as it contains 54 digital I/O pins (15 provides PWM output) , 16 analog inputs, 4 UARTs and a flash memory of 256KB memory for big sketches. This makes it perfect for a 3D printer or robot project. The processor operates at 16MHz. The Mega also have a USB input which is combined data transfer/power supply. This makes it possible to power the board with just a computer, or any USB power supply.[1]

### 3.7.1 Sensors

#### DHT22 - Humidity/Temperature sensor

The DHT22 (AM2302) is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air[2].

---

[1] "Arduino Mega 2560 Rev3 - Arduino Store." https://store.arduino.cc/usa/arduino-mega-2560-rev3. Öppnades 7 maj. 2018.

[2] "DHT22 Temperature-Humidity Sensor - Waveshare Wiki." 2 aug.. 2017,

The sensor needs 2 seconds to boot up and become stable.

The DHT22 measures relative humidity. Relative humidity is the amount of water vapor in air vs. the saturation point of water vapor in air. At the saturation point, water vapor starts to condense and accumulate on surfaces forming dew.

**SI1145 - UV index / IR Sensor**

SI1145 calculates the UV index. The sensor is not actually measuring the UV light but instead it measures visible and IR light from the sun. UV light is light from the ultraviolet spectrum[3]. The UV light is not visible for the human eyes but it especially affects both skin and eyes. The sensor is digital and works over I2C. I2C is a synchronous and serial computer bus which is widely used when connecting low speed integrated circuits(IC) to processors and microcontrollers.[4]

**HC-05/06 - Bluetooth Module**

The bluetooth HC-05/06 is a bluetooth module that can be connected to an arduino device. The bluetooth module makes it possible to archive serial data transmission wireless and the interface is set to serial which makes it easy to use and implement in projects. The module operates on the 2,4GHz ISM (industrial, scientific, medical) frequency band[5]. The same as e.g. NFC, WiFi networks, Microwave ovens etc. The device uses Bluetooth 2.0+EDR standard. The standard are specification for short-range communication. The EDR stands for Enhanced Data Rate, which is a faster PSK modulation scheme which makes it possible to transmit data 2 or 3 times faster than previous versions of bluetooth[6]. The

---

https://www.waveshare.com/wiki/DHT22_Temperature-Humidity_Sensor. Öppnades 7 maj. 2018.

[3] "Adafruit SI1145 Breakout Board - UV index / IR / Visible Sensor." 21 mars. 2014, https://learn.adafruit.com/adafruit-si1145-breakout-board-uv-ir-visible-sensor/overview. Öppnades 7 maj. 2018.

[4] "I2C Bus: I2C - What's That?." https://www.i2c-bus.org/. Öppnades 7 maj. 2018.

[5] "Bluetooth Transceiver Module HC-06 - Wiki." 20 mars. 2017, http://wiki.sunfounder.cc/index.php?title=Bluetooth_Transceiver_Module_HC-06. Öppnades 7 maj. 2018.

[6] "What is Bluetooth 2.0+EDR? - Definition from ... - Whatis Techtarget." http://whatis.techtarget.com/definition/Bluetooth-20EDR. Öppnades 7 maj. 2018.

specifications means that the HC-05/06 has a maximum transmission rate at 2.1Mbit/s. The signal transmit time is set to 0.5 seconds intervals for different devices so the workload of the chip is reduced.

The module has the following default settings:

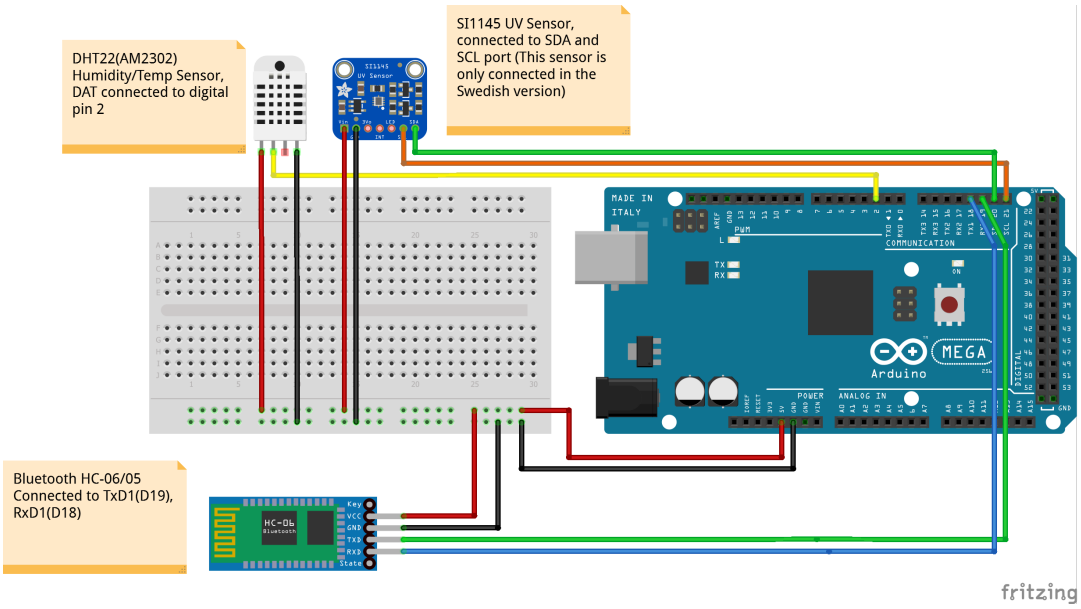
Baud-rate: 9600 N81(Common data rate config with: no parity bit, 8 data bits and 1 stop bit)

Standard-pin: 1234

Standard name: HC-05/06

The HC-06 can only be used as a slave-device compared to the other version HC-05 that can be either master or slave.

### 3.7.2 Circuit schema




## 3.8 The Ev3

### 3.8.1 Sensor

We will use:

- **Color Sensor:**

Description:

The digital EV3 Color Sensor distinguishes between seven different colors and can also detect the absence of color. It also serves as a light sensor by detecting light intensities. Students can build color-sorting and line-following robots, experiment with light reflection of different colors, and gain experience with a technology that is widely used in industries such as recycling, agriculture, and packaging.

Why we need this:

To helps the robot track the black line in warehouse map, follow those line to reach the storage in warehouse.

- **Gyro Sensor:**

Description:

The digital EV3 Gyro Sensor measures the robot' s rotational motion and changes in its orientation. Students can measure angles, create balancing robots, and explore the technology that powers a variety of real-world tools such as Segway®, navigation systems, and game controllers.

Why we need this:

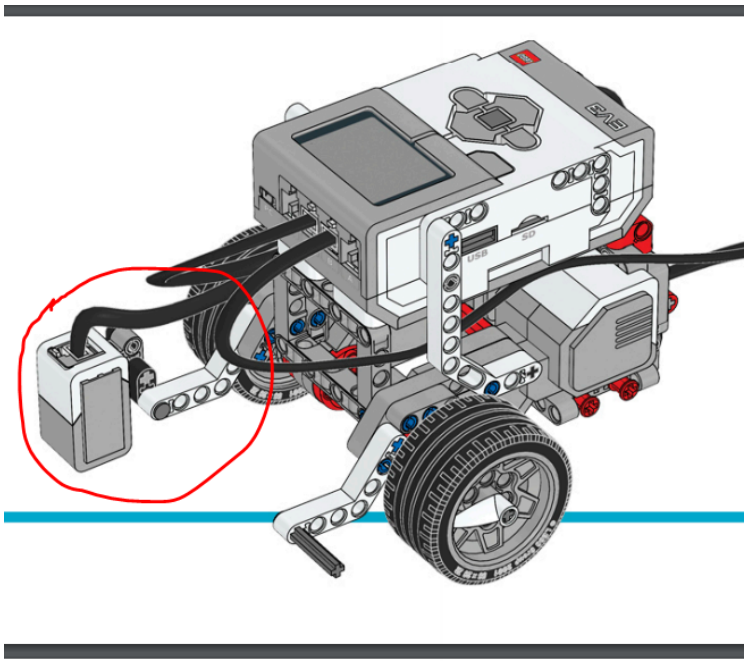
To helps the robot turn when it comes to the crossing

### 3.8.2 Robot Model

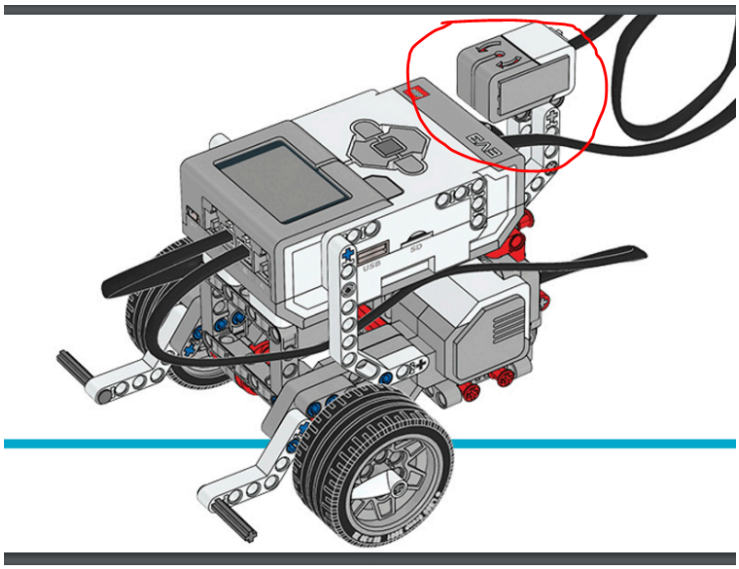
We will use the default model in the instruction that come with the EV3 box.

The robot sensors will be place like following:

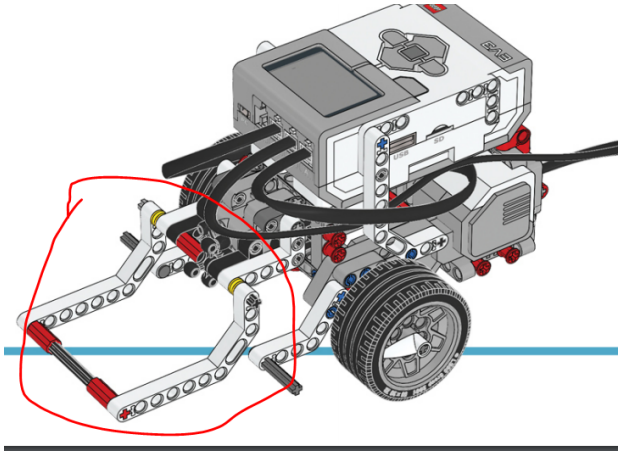
Color Sensor:



Gyro Sensor:



Lifting Arm:



### 3.8.3 Robot Movement:

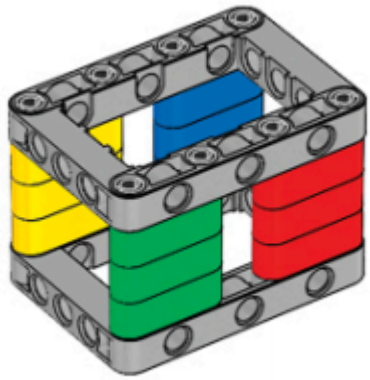
1. By WASD (NOT IMPLEMENTED?)

The robot will be manually control by the user through web interface.

2. By Command

The user will command the robot (ex: take the package from storage A out of warehouse), then the robot will run automatically and perform the necessary action to complete that command.

Package Model:



Package will be similar to this one, but with more open space for lifting arm of the robot going through.

## Instructions how to build the main robot

This is the instructions you may follow to build our robot design. The arm and the package

are differs from the official Lego instruction below (see Instructions how to build the robot arm and the package model). Please notice that we only use 2 sensors, which are color sensor and gyro sensor. This is the only sensors you need to run our project. Adding more sensor are unnecessary in this case.

https://education.lego.com/en-us/support/mindstorms-ev3/building-instructions

### 3.8.4 Instructions how to build the robot arm and the package model

This is the detailed instruction how to build our version of **the arm** and **the package**. Notice that: You can build your own design but we strongly recommend to use our design for this project. This is because our design is fully tested designed for our warehouse so the robot will run more effective and more stable

https://docs.google.com/document/d/1GMIpcG2im8pSA-D3m26qiw2eaMByLk4dNJzOWBpwbGc/edit

## 3.9 Warehouse

### 3.9.1 Requirements:

- The EV3 can reach all *inputs, outputs,* and *storage units*.
- An *input* is a location where items are delivered to the *warehouse.*
- An *output* is a location where items are taken out of the *warehouse*.
- A *storage unit* stores the items in the *warehouse*.
    - A *storage unit* consists of *shelfs* where items may be stored.
    - A *storage unit* must have at least one *shelf*.
    - There is no upper limit to the number of *shelves* a *storage unit* can have as long as the EV3 can reach all the shelves.
    - To have multiple shelves per storage unit is a stretch goal rather than a requirement.

The warehouse should be simple; one spot for *input*, *output*, and up to four

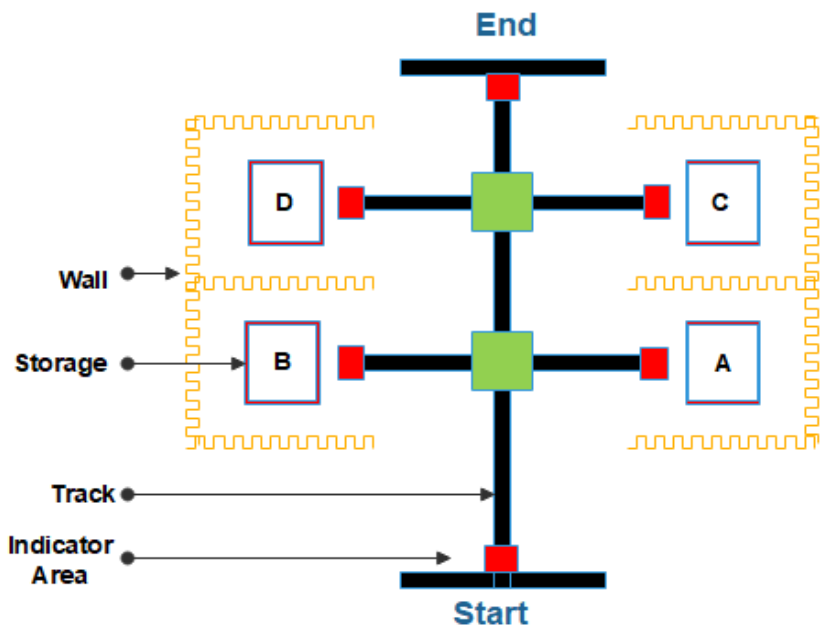
*storage units*. No need to build a complex warehouse for this proof of concepts.

Example design of a warehouse:

```
  In
++++|++++
+A--|--B+
++++|++++
+C--|--D+
++++|++++
 Out
```

Where `A`, B, C, `D` are storage, "`In`" is the warehouses input location and "`Out`" is the warehouse' s output location.
Each storage location may be able to hold one or more items.
The project is not about creating an efficient warehouse design, therefore we should keep it simple to build and easy for the EV3 to travers.
The solution will be scalable meaning we implement a warehouse with 4 slots in the prototype, but it should be possible to extend it to 8, 12, 16, 20 and so on.
The "+" character indicates the walls of the warehouse and the "-|" character indicates the possible route of the robot.

### 3.9.2 Warehouse map in practical




Robot will run following the black track.

The indicator Area will help robot know where it is, and also what should it do.

In this picture:

- the blue area indicates the Start and End titles.
- the red area indicates the storage stop
- the green area indicates the intersect of the track

The basic workflow of the robot would be:

- From start position, robot is given a command to transfer the package
- The robot will set up the path
- It will start moving by tracking the edge of the black line
- When it meets green area, it will stop and based on the command, it will turn to left, right or go forward

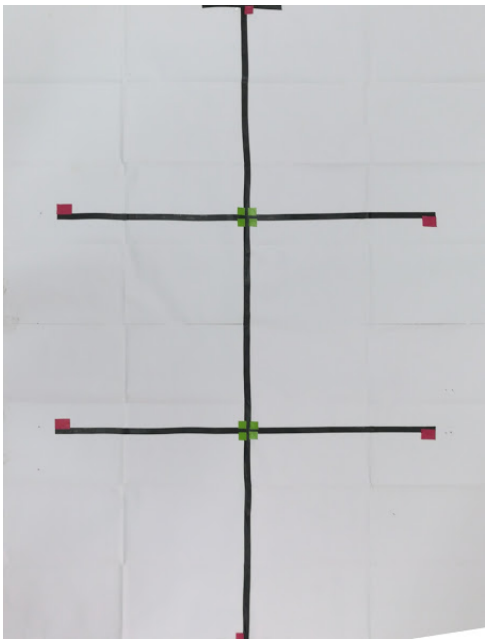
  eg:

  command is "take package from A to end", it will turn right and head to A storage.

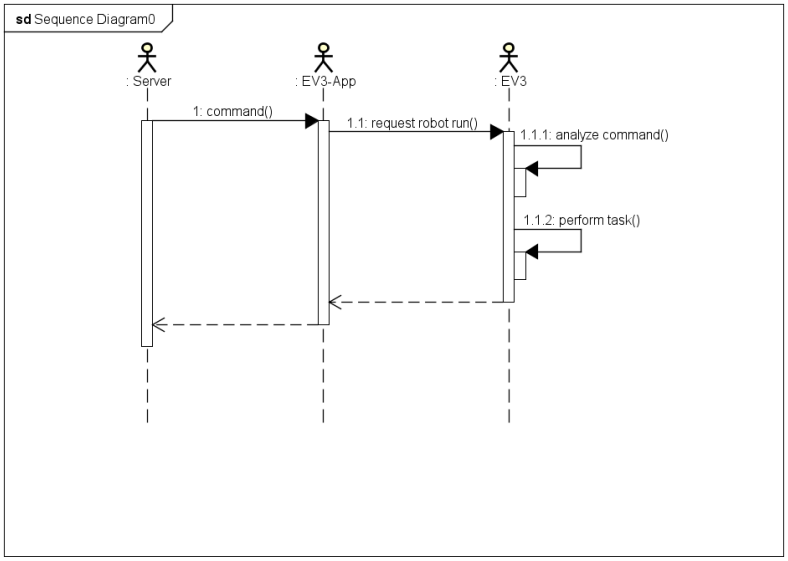
  command is "take package from D to end", it will turn go forward, and turn left at next cross.
- When it meets red area, it will stop and pick the package

- When it meets the blue area, it will drop the package and head to the start title

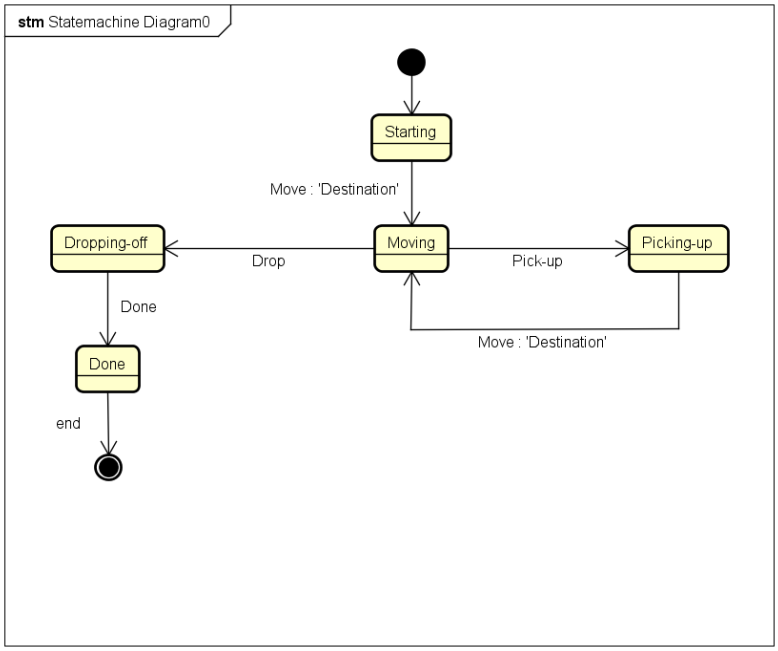
### 3.9.3 Real photo of the warehouse map



Sequence diagram for the ev3 communication part




Robot state diagram




## 3.10 Server, Database, and UI

### 3.10.1 System Requirements

From the specifications given by the customer we can see that the Back-End and UI of the system require the following.

1. A way of collecting, and displaying the sensor data collected from the warehouse and alarm the user if these are not within the allowed range
2. A way of remotely giving commands to the robot
3. Provide a real time graphical representation of the warehouse to the user
4. The user is to be able to prioritize different storage areas and operations[7]

### 3.10.3 Technical Specification

From the System Requirements we can derive a Technical Specification.

---

7 https://docs.google.com/presentation/d/19NMczo2IrNMs4sfV54ohA9TQfNidy9N_rgY6DOOVWz8/edit?usp=sharing slide 7 and 8

## User Interface

The User Interface has two main functions:

1. To give the User the ability to give orders to the EV3, and
2. to inspect the warehouse.

After a discussion with prof. Arnold Pears he made it clear that no Graphics were required for us to get a passing grade in the course. Thus, we decided we would do the most basic interface possible so that it would not take up more time to develop than absolutely necessary.

The UI will be a Command Line Interface in the Console and will contain the following Screens.

### Welcome / Start

The first screen seen by the user, will have two options:

1. Inspect Warehouse
2. Jobs
3. Exit

### Inspect Warehouse

The screen where the user can look through the warehouse and find the robot and how many items are stored and where. This screen will display

1. A graphical representation of the Warehouse ( "ascii art" -esuq)
    a. A map of the Warehouse
    b. The location of the EV3
    c. Number of Items in the Storage Units
    d. Name of the Storage Units
2. Arduino Sensors
    a. Current sensor readings
    b. Threshold for acceptable states of warehouse

The Options given in this screen are

1. Change source temp./light threshold in Warehouse
    a. Selecting this will prompt the user to input a new value for the chosen attribute. After a value has been given the user is returned to this screen.
2. Go back to previous Screen

### Jobs

This screen is the users interface with the EV3. Here here the user can give orders to the EV3, find it, view what the EV3 is doing, and where it is going.

The Screen will Display:

1. EV3' s current job
2. EV3' s status, what is it doing right now?
3. The prioritized Storage Unit
4. How many more items can be stored in the warehouse

The options given:

1. New job
    a. From where
    b. To where
2. Cancel job (how would this work?)
3. Choose prioritized storage unit
4. Go back to previous Screen

Sequence Diagram

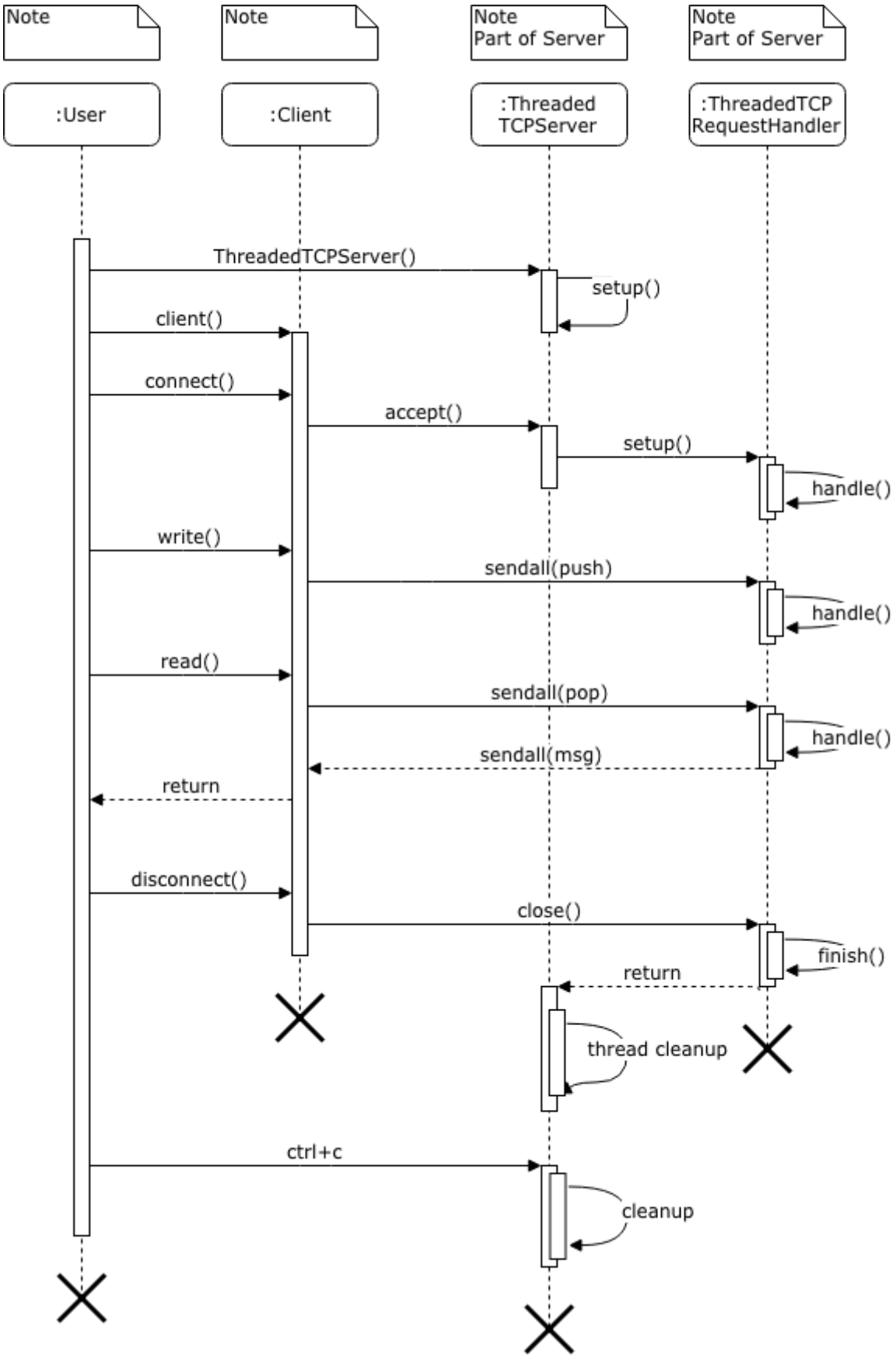
## Server

The Server' s main task is to allow communication between the different parts of the System. It uses a Message Queue to keep track of the order of the Messages passed through the system. Every Client that connects to the Server has a Type that determines what Messages the Server sends to them. The Server spawns a new thread for each Client that connects.

Each part of the System has a Client interface that abstracts away most of the Client-Server communication into four functions and one class. connect(), disconnect(), write(recipient, message), and read().

| | |
|---|---|
| client(host, port, type) | Creates a new client object. The values host and port are used to find the server, and the value type is a string representation of what kind of client this is supposed to be. |
| connect() | Connects the client to the Server using host and port specified when creating the client object. |
| disconnect() | Closes the socket to the Server in a correct manner. |
| read() | Retrieves a messages for this client from the server or will get a message stating that it' s reached the end of the queue. |
| write(recipient, message) | Sends the message to the server which puts it in the message queue and keeps it there until the recipient(s) reads the message from the server. |

Sequence Diagram



### 3.10.4 Database

MariaDB version 10.2.14

The database stores the state of the warehouse. And the status of jobs. Each job has multiple transactions based on the positional status of the robot. The database also has the sensor information from the arduino viz; Temperature, Humidity, etc.

The database here is named `automated_warehouse_management` and 3 tables which are:

- storage_unit
- sensor_data
- ev3_robot

- The storage_unit table consists of the names, number of items available currently and the maximum capacity of the respective storage units.
- The sensor_data table consists of different sensor data which can be used to determine the functional state of the warehouse at a given time.

  The table has Humidity, Temperature and Luminance values.

- The ev3_robot table has the Job ID, Status and Position of the robot respective to the job. The status of the robot varies for every transaction recorded viz;
  - Started
  - Moving
  - Picked Up
  - Delivering
  - Done

- Based on the status received from the robot on a job transaction, a trigger is employed to auto increment the job_id.

- Python MySQLdb module will be employed to connect the database to the server.

## 3.11 Communication

Here we will explain how our communication protocols are designed. All messages are formatted as json-strings.

### 3.11.1 Robot-Server communications

The robot sends the server regular update regarding it' s position, and job status. The message is passed to the database for updating the representation of the robot. The message is sent as a json string in the form of:

EV3 status message: { 'push' :True, 'to_db' :True, 'ev3_status' :True, 'status' :*N*, 'position' :*M*}

Where *N* and *M*, are strings representing the status and position. The robot sends this message to the server every time the robot moves.

### 3.11.2 Server-Robot communication

When the user creates a command for the robot to follow a message containing the pick-up and drop-off location.

EV3 job message: { 'push' :True, 'to_ev3' :True, 'ev3_job' :True 'to' :*X*, 'from' :*Y*}

Where *X* and *Y* represent some location in the warehouse, and they are not the same location.

### 3.11.3 Robot-Robot communication

There is no real robot to robot communication in this system. The ev3-app handles the coordination between the robots, making sure it does not give orders to the different robots that would put them both in the same location or colliding.

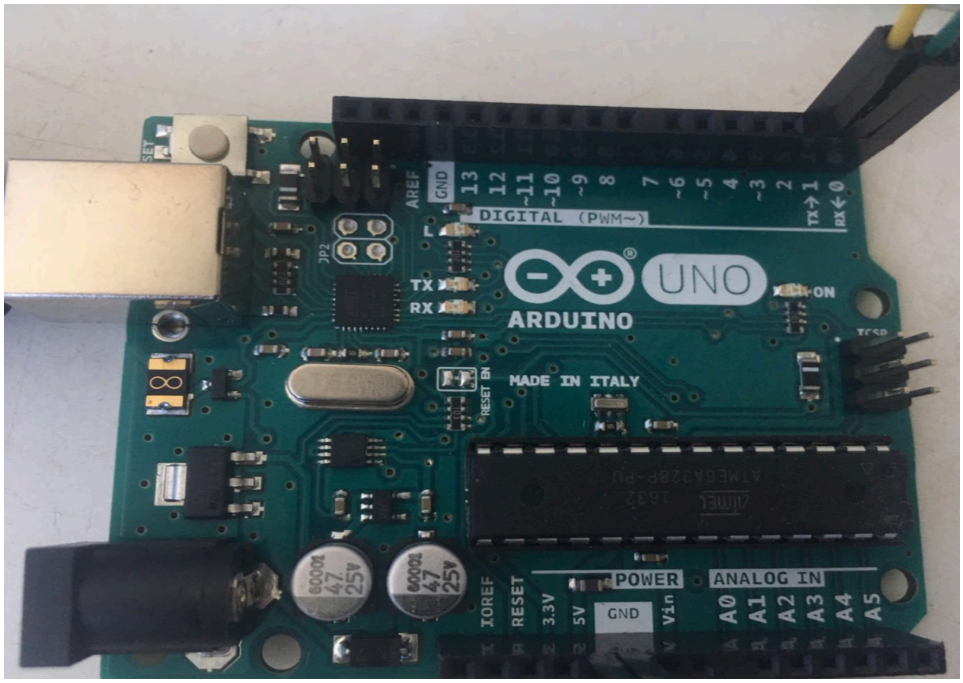
# 4 System Implementation

Because this report is written by team Turku, and only responsible for Arduino, so in this part we focus on Arduino mainly.

## 4.1. Components introduction

### 4.1.1 Arduino

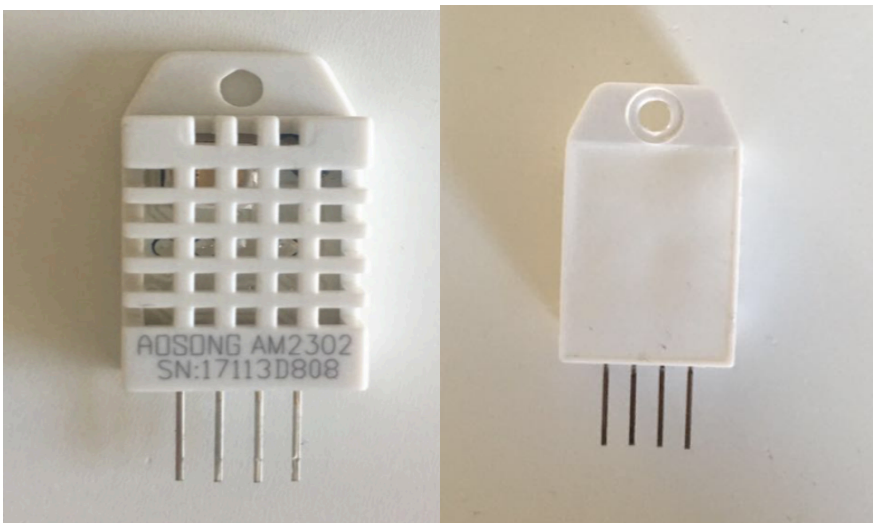
The Arduino Mega 2560 is a master development board based on the ATmega2560. The Arduino Mega2560 is a core circuit board with a USB interface, with 54 digital inputs and outputs, it is suitable for designs that require a large number of IO interfaces. And the processor core is the ATmega2560, which has 54 digital input/output ports, 16 analog inputs, 4 UART interfaces, a 16MHz crystal oscillator, a USB port, a power socket, an ICSP header, and a reset button. There are all resources on the board supporting one master board. It can automatically select 3 power supply modes: external DC power supply through the power socket; battery connected to the power connector GND and VIN pins; USB interface DC power supply.



### 4.1.2 Sensor AM2302

**Basic Introduction**

AM2302 Humidity Capacitive Digital Temperature and Humidity Module is a temperature and humidity composite sensor that contains a calibrated digital signal output. It applies a dedicated digital module acquisition technology and temperature and humidity sensing technology to ensure that the product has high reliability and excellent long-term stability.




**Product parameters:**

Size: 40*23mm

Weight: 4g

Voltage: 5V

Port: Digital Bidirectional Single Bus

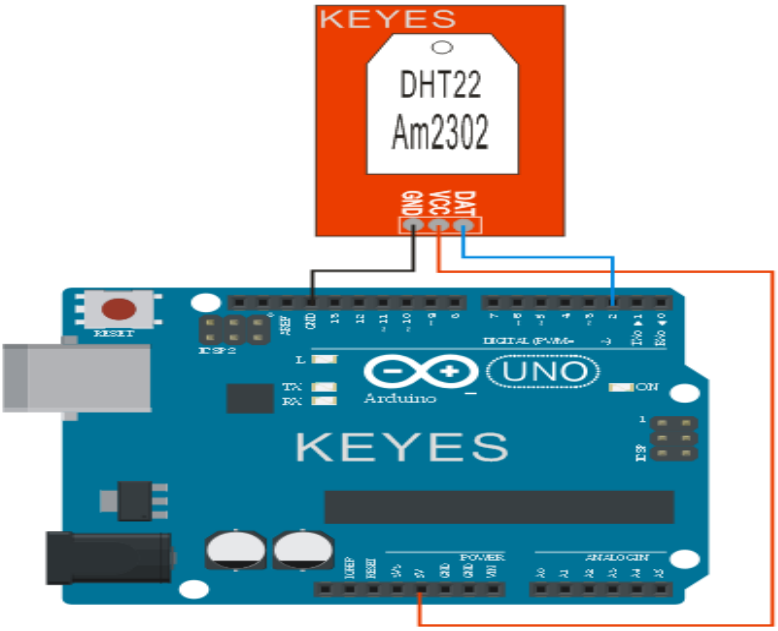
Temperature range: -40-80°C ±0.5°C

Humidity range: 20-90%RH ±2%RH

Platform: Arduino, Microcontroller

**Connection to the Arduino board:**

The "VCC" of the module is terminated with a +5V output, the "GND" is terminated with a GND, and the "DAT" is terminated with a digital port with pin number 2 (of course, this can define the digital pin by itself).

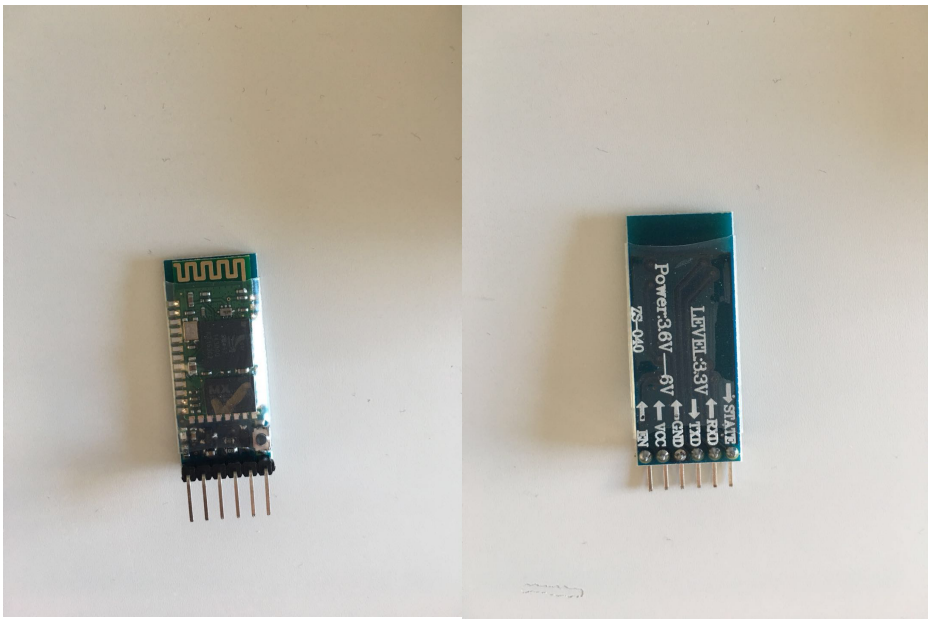



### 4.1.3 HC-05 Bluetooth Module

**Introduction**

The bluetooth HC-05 is a bluetooth module that can be connected to an arduino device. The bluetooth module makes it possible to achieve serial data transmission wireless and the interface is set to serial which makes it easy to implement in projects.

The module operates on the 2,4GHz ISM (industrial, scientific, medical) frequency band, such as NFC, WiFi networks, Microwave ovens etc. The device uses Bluetooth 2.0+EDR standard. This standard is specification for short-range communication, the EDR stands for Enhanced Data Rate, which is a faster PSK modulation scheme which makes it possible to transmit data 2 or 3 times faster than previous versions of bluetooth. The specifications mean that the HC-05 has a maximum transmission rate at 2.1Mbit/s. The signal transmit time is set to 0.5 seconds intervals for different devices so the workload of the chip is reduced.




The module has the following default settings:

Baud-rate: 9600 (Common data rate config with: no parity bit, 8 data bits and 1 stop bit)

Standard-pin: 1234

Standard name: HC-05

### 4.2. System integrity

#### 4.2.1 Hardware requirements:

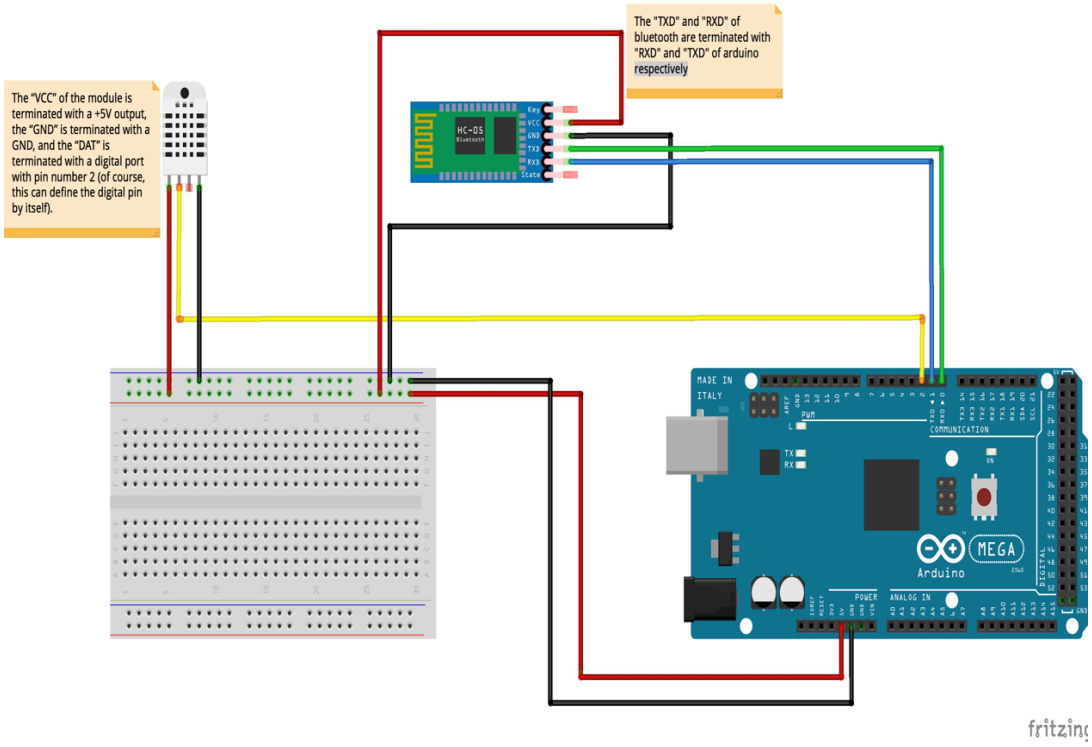
Arduino Controller × 1

USB data cable × 1

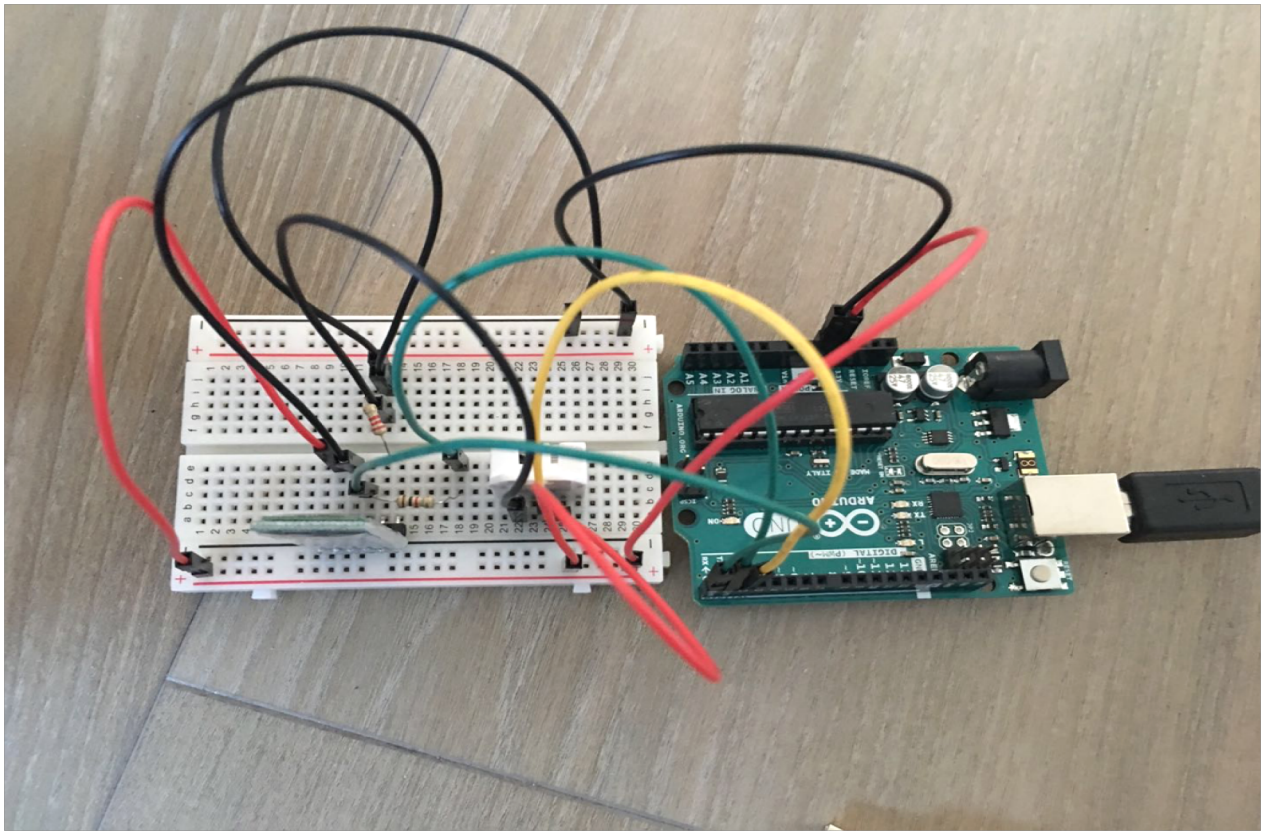
DHT22 module × 1

HC-05 module× 1

#### 4.2.2 Overall Circuit Schema



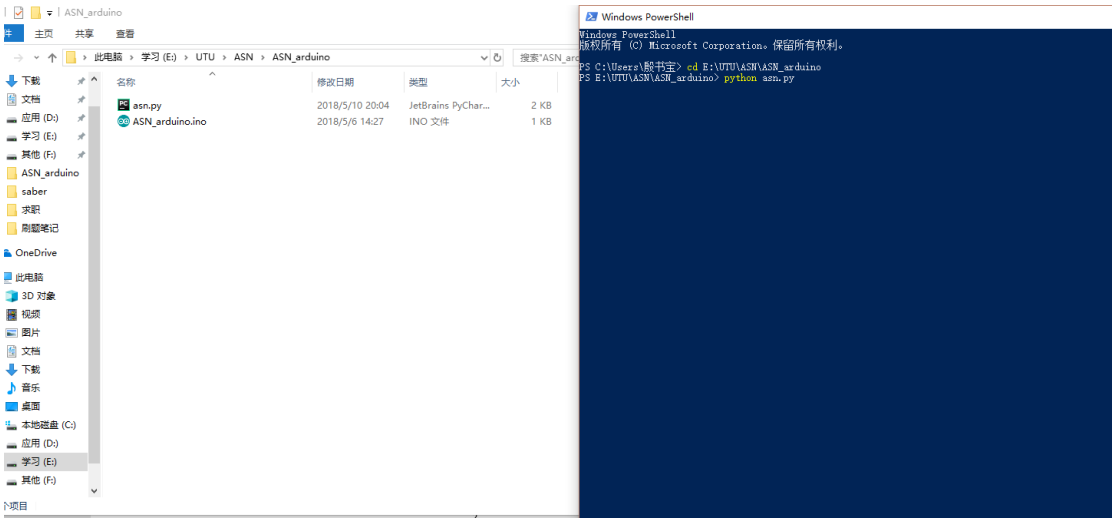

Circuit schema



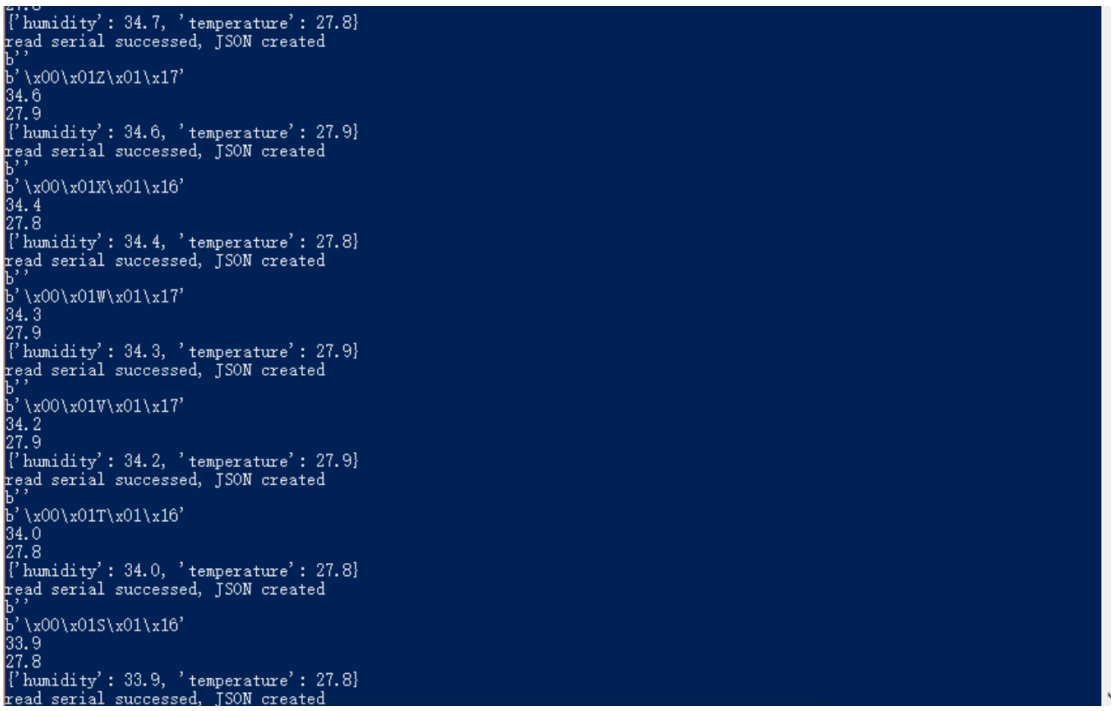
Physical circuit schema

## 4.3 Working flow

First, connect the wires each component according to the method above, connect HC-05 Bluetooth to the server by using UART, configure on the computer (because sever is not finished by other team members, so we test in local), remember to check the COM port consistent with the COM port of python file, and run the python file, the server can receive the values of sensor.




Then we can see the value of humidity and temperature on the terminal, and note that JSON file is success created.




After using my mouth to blow hot air on the sensor, the temperature and humidity have changed obviously.

```
{'humidity': 33.8, 'temperature': 27.8}
read serial successful, JSON created
b''
b'\x00\x01R\x01\x17'
33.8
27.9
{'humidity': 33.8, 'temperature': 27.9}
read serial successful, JSON created
b''
b'\x00\x01Q\x01\x17'
33.7
27.9
{'humidity': 33.7, 'temperature': 27.9}
read serial successful, JSON created
b''
b'\x00\x01\x9a\x01\x17'
41.0
27.9
{'humidity': 41.0, 'temperature': 27.9}
read serial successful, JSON created
b'\x00\x02z\x01\x1a'
63.4
28.2
{'humidity': 63.4, 'temperature': 28.2}
read serial successful, JSON created
b''
b'\x00\x03x\x01\x1e'
88.8
28.6
{'humidity': 88.8, 'temperature': 28.6}
read serial successful, JSON created
b''
b'\x00\x03\xbe\x01\x1f'
95.8
28.7
{'humidity': 95.8, 'temperature': 28.7}
read serial successful, JSON created
b''
b'\x00\x03\xbb\x01\x1c'
95.5
28.4
{'humidity': 95.5, 'temperature': 28.4}
read serial successful, JSON created
```

And you can see test.json in the folder.

| 名称 | 修改日期 | 类型 | 大小 |
|---|---|---|---|
| asn.py | 2018/5/10 20:04 | JetBrains PyChar... | 2 KB |
| ASN_arduino.ino | 2018/5/6 14:27 | INO 文件 | 1 KB |
| test.json | 2018/5/10 20:20 | JSON 文件 | 1 KB |

open the test.json, you can see the current value of humidity and temperature.

test.json - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
{"humidity": 95.5, "temperature": 28.4}
```

Note:

```
Serial.write(0x00);
Serial.write(highByte(h));
Serial.write(lowByte(h));
Serial.write(highByte(t));
Serial.write(lowByte(t));
delay(2000);
```

We totally send 5 bytes data from Arduino write to the serial, 0x00 is the start byte, second and third bytes are the value of humidity, while fourth and fifth bytes are the value of temperature.

```
data = ser.readline()
print(data)

if len(data) == 5:
    humidity = int.from_bytes(data[1:3], byteorder='big') / 10
    print(humidity)

    tem = int.from_bytes(data[3:5], byteorder='big') / 10
    print(tem)

    dict_json = {}
    dict_json['humidity'] = humidity
    dict_json['temperature'] = tem
```

In python file, read from corresponding bytes for humidity and temperature, convert these values to integer, then transform to json format.

# 5 Test Result

In practical Team Turku and Team HUST finished their job one week before the final presentation. However, Team Uppsala finish their job till the morning in the day of final presentation. So in final presentation, Team HUST and Team Turku showed their work of Robot and Arduino.

Because Team Uppsala have not finished their Internet Communication, Team Turku had to finish the Sensor – Arduino – PC – Bluetooth – EV3 communication themselves. And by studying Team HUST' s EV3 code, team Turku also finish the warehouse map and finish the control of EV3, which is shown in the final presentation.

# 6 Summary

In this project, we designed a system with the function of sensor network, robot and server. The final presentation and demonstration showed we had done a good coordination with two different countries with a heavy burden of time difference. This this project, Turku team finished the function of sensor network

and script to store the data on the local terminal. Also, we finished the demo to use the sensor value to control the robot. The process of the design, communication and team work was rewarding.

Meanwhile for all of us, it is our first time participated in the global team work, which teaches us a lot on how to cooperate with teammates we has never met yet. It is a very good experience.

# 7 Code

https://github.com/TimSvensson/gsdp2018-team5