IoT Project: Height / capacity monitor

Grupo K:
Tim Reiprich
Tegshigzugder Otgonbayar
Luca Maltagliati

January 19, 2020

This documentation will be updated until the final delivery.

Contents

1 Introduction

In the project of the course Internet of Things we are to implement a device, that could be somewhat used in an environment. For this project we chose from the possible project ideas the Height / capacity monitor. We picked this idea, because it was interesting to see how it could be implemented and used in the field. Especially as we could think of certain sceneries, where it can be used efficiently such as, in household, for example where there are children to have such a sensor and turn the water off or even send a notification. In industrial area probably for indicators of highly valuable liquid. — must be reformulated (maybe more industrial than household with children)

Our task was to create this device and also make it possible to connect with other For an inexpensive implementation we worked with the ESP8266. During the development we used the Arduino IDE and to easily create an user interface Node-Red was used. **Context and general summary of the operation of the system**

2 Context and general description of the device

Our project focuses on the measurement of containers holding some kind of liquid. Based on the volume of these containers alarms can be sent to the users to notify them about the amount. To accomplish this, a sensor will be used, that can determine the height of the liquid in the container. Furthermore an upper and a lower threshold can be set by the user. If one of these is exceeded, an alarm in form of a LED at the sensor and a virtual LED in Node-Red goes off.

2.1 Hardware scheme

The basic scheme of system hardware is described as follows: the main components are a computer as central server, the sensors and a web-page. The sensors must be able to connect to the server via a WiFi-connection and for this we assume that the connection is robust and will be available in the given environment. The homepage is hosted on the central server and functions as the main way to interact with the system for the enduser. In theory it wouldn't be difficult to expand the system to function with multiple height sensors as seen in figure 1. But due to restrictions in available components we implemented only one sensor communicating with the central server in the final project.

2.2 Sensor

We used an ultrasonic sensor to measure the level of the liquid. The one we used, is called Ultrasonic Sensor Module HC-SR-04 by Robokart. We want to choose this approach because it is one of the best ways to sense proximity and detect levels with high reliability. The sensor is connected to a microcontroller that collects the data and sends them to the central server via WiFi.

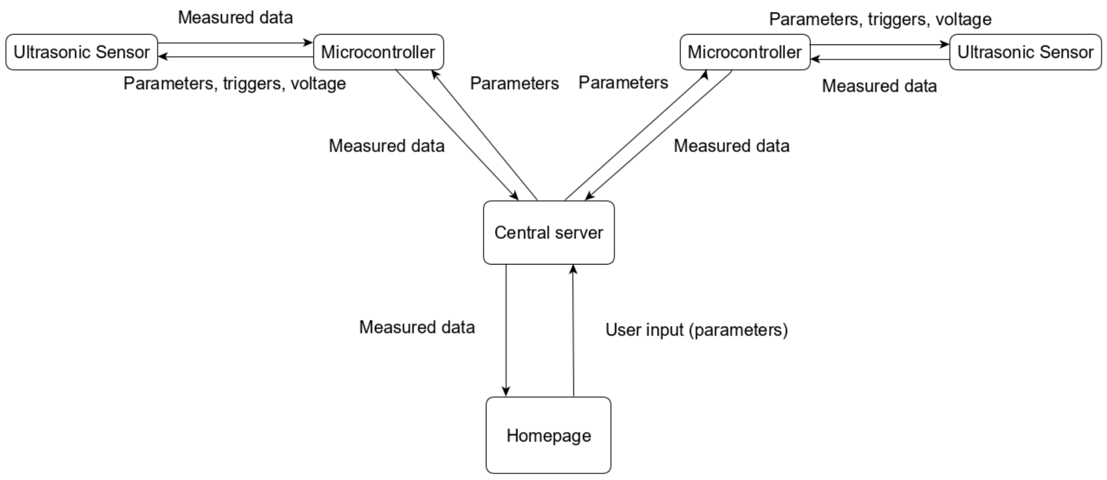


Figure 1: Hardware scheme

Every container will have a microcontroller that is connected to at least one sensor (see figure 2). Certain specified heights of minimal and maximal volumes can be set dynamically, depending on various factors, like the position of the container, the probability of it being used, the liquid type and as such.

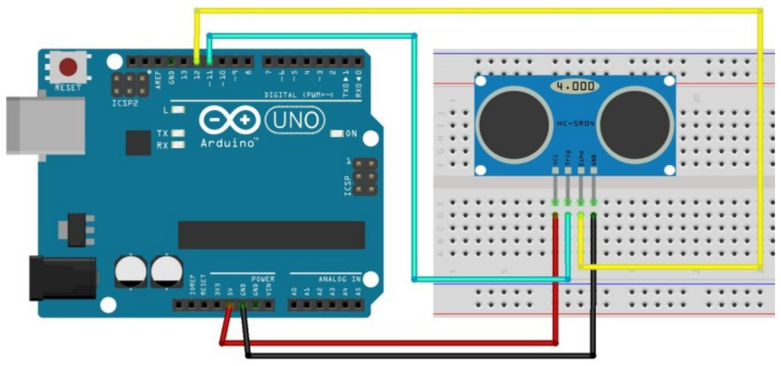


Figure 2: Example of a sensor connected to a microcontroller

2.3 Central server

We assume a LAN or WLAN network to which the controllers and central server are connected. Through this the microcontrollers will be able to communicate with the central server by sending the measurements of the liquid heights continuously. The central server will perform checks on the thresholds curerntly set by the user and, if needed, will show an alarm on the homepage and send the alarm signal to the microcontroller connected to the sensor, so the alarm in form of the connected connected LEDs will be shown as well. This procedure and the corresponding homepage was configured using Node-Red.

3 Steps of the project

The first steps of the project consisted of research about similar projects and identifying platforms, programming languages and frameworks, choosing the most suitable ones for our case. Furthermore, the connection between devices plays a crucial role, so we needed to know what kind of protocols are needed. During the next phase we planned the structure of the system and began to get familiar with the tools we wanted to use. After the planning was done we implemented the system and tested it afterwards to remove possibly found errors.

3.1 List of used components

- NodeMCU DevKit v1.0 development board (ESP8266) - we will use the microcontroller connected to sensor. It collects and sends data to broker and receives parameters to light up the warning LEDs correctly.
- Ultrasonic proximity sensor (digital / analog output module) - the main sensor to measure the distance between liquid and sensor
- power source for the microcontroller and sensor (batteries, solar panel or a similar source)
- Small materials:
  - cables - to connect sensor, microcontroller and LEDs
  - breadboard - to connect sensor, microcontroller and LEDs
  - LEDs - to signal that level of liquid has exceeded height thresholds set by the user

3.2 Wiring scheme

During the implementation of the system we used the setup seen in figure 3. We used batteries to power the system which makes it more flexible to use but results in downtimes when the batteries have to be changed. If a more stable system is desired the batteries can be exchanged with a permanent connection to a power socket.

3.3 Flow chart of system functionality

Following the wiring scheme the execution flow was planned. As seen in figure 4 the parameters regarding the thresholds and the height of the basket the sensor

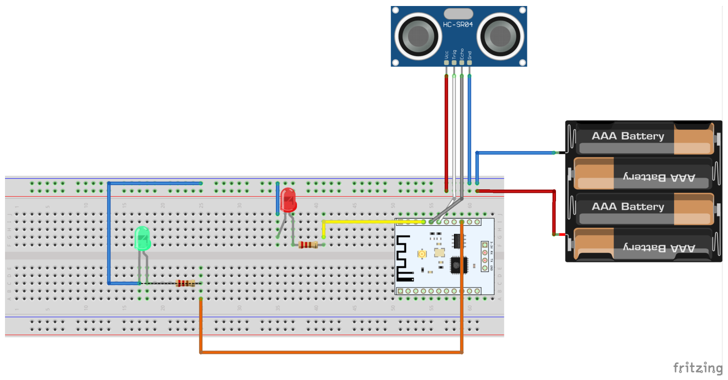


Figure 3: Hardware scheme

is used in have to be set by the user at the beginning. Afterwards the system enters a loop until its execution is stopped through a disconnect from the MQTT broker or a loss power supply.

3.4 Node-Red

This section covers the parts of the project running on the central server which mainly includes the homepage created with Node-Red. The flow we created for this project can be seen in figure 5 and the simple user interface in figure 6. The flow consists of 4 input elements located on the left. Located in the middle of the flow there are nodes transforming the input data and on the right all the output nodes are located.

3.4.1 Input nodes

3 out of the 4 input nodes are simple text nodes through which the user can set the basket height, lower and higher threshold. All 3 can be changed at any point even if the system is running. This may be needed in cases like the liquid in the basket changed and due to a higher density the basket can hold less than before, the sensor is used in a new basket or similar use cases. The basket height has to be set because the used ultrasonic sensor sometimes measures faulty values. So the basket height is used to filter them and prevent the microcontroller from sending them to the MQTT broker.

The other input node receives MQTT messages from the MQTT broker. It is subscribed to the topic “distance” to which the microcontroller publishes the measured distance.

3.4.2 Transforming and output nodes

The function node called “Thresholds” checks if the user given thresholds are valid. This means it checks if the lower threshold is lower or equal to the higher trehshold and the other way around. If this is the case the new values for lower and higher threshold are stored in`conext.low` and `context.high`. Furthermore the node returns `{payload: {upper: context.high, lower: context.low}}` which is used by the two connected text nodes on the right side to show the user the currently set thresholds on the homepage.

The function node "Threshold Check" does the same checks as the previously discussed node and also stores the current threshold values in a similar manner. Additionally it also stores the last measured distance and uses it to check if one of the two thresholds is surpassed. In the end the object `{payload: {highLed: <boolean>, lowLed: <boolean>, basketHeight: context.basket}}` is returned where the property highLed or respectively lowLed is set to "true" if the corresponding threshold is surpassed. The last property always contains the currently set basket height. This is needed because the ultrasonic sensor returns abnormal values at some times. So we use the `basketHeight` to filter out all distance measurements higher than the height of the basket the sensor is used in as these are impossible and definitely false. Afterwards this object is published to the topic "threshold" as a JSON using MQTT. The microcontroller is subscribed to this topic and uses the received values to turn its connected LEDs on or off and to filter abnormal measured distances.

Besides that the returned object of the node "Threshold Check" is also used as input for the function node called "filterLow". It simplifies the input and returns the object `{payload: "high"|"low"|"between"}` where the value is chosen depending on the object given as input. We used this so we can simply check for received string in the LED nodes afterwards in the flow to determine if the LEDs have be turned off or on.

And finally the two output nodes on the top are used to show the measured distances. The text node just shows the last measured distance in centimeters. The node called "Distance" is a graph of the measured distances which enables the user to see the tendencies in the liquid level and act before the critical thresholds are exceeded.

3.4.3 User interface

The user interface (figure 6) is structured in 3 parts. On the left the current status of the system is shown meaning the current parameters of basketheight, lower and higher treshold. Also it contains the 3 status LEDs signalizing if one of the thresholds is surpassed or if the system is in a non-problematic state (shown by the green LED names "normal" being turned on).

In the middle the measurements are shown. This part consists of the last mesured distance and a graph showing the changes during the last measurements. These values can only be seen if a microcontroller with an ultrasonic sensor is publishing its data. Otherwise the graph is blank and the message `desconnectado` is shown instead of the last measurement.

The right-most part contains all nodes that allow user input. Through these the user can set the basket height, lower and higher threshold. This can happen dynamically at any time. The changes will be published to the MQTT broker and therefore the microcontroller everytime one of the values is changed or after a new measurement arrives.

3.5 Micro controller programming

3.5.1 Source file

All of the code is contained in the source file `project.ino`, compiled and sent to the `ESP8266WiFi` microcontroller using `Arduino IDE`.

3.5.2 Communication with the sensor

The logic of the distance measurement is implemented in the `loop()` function; every half a second, the measurement is done and published on the `MQTT` topic. Since there's no need to publish additional information, the data is published as a simple plain text, containing only the number with the measurement or a string with an error message.

We are using an ultrasonic sensor. First we ensure the sensor is at low, then we send a 10 milliseconds-long pulse and measure the time passed.

The function used to interact with the sensor is `digitalWrite(trigPin,flag)`, where `trigPin` is the identifying number pin on the microcontroller to which the trigger Pin of the sensor is connected to (see the hardware schema), and `flag` can be either `LOW` or `HIGH`, depending if we want to trigger on or off the ultrasonic sensor.

The distance is then calculated with the formula:

$$Distance = \frac{Time/2}{29.1} \tag{1}$$

3.5.3 Communication with the base station

The `callback()` function is invoked every time new data is published on the topic. In particular, it is programmed to receive the `JSON` payload described in the previous section; we used the `JSON` parsing library `ArduinoJson.h` to be able to deserialize it. The payload is deserialized and the corresponding LEDs are turned off or on, basing on threshold trespassing information. Also the max height of the basket can be set this way.

We also check the validity of the received `JSON`, debugging an error message in case of syntax errors in the payload.

3.5.4 Other functions necessary for device operation

The functions `reconnect()` and `setup_wifi()` check and/or establish the connection to the WiFi network. `reconnect()` is always called at the start of every `loop()` call, to ensure the micro controller is always connected.

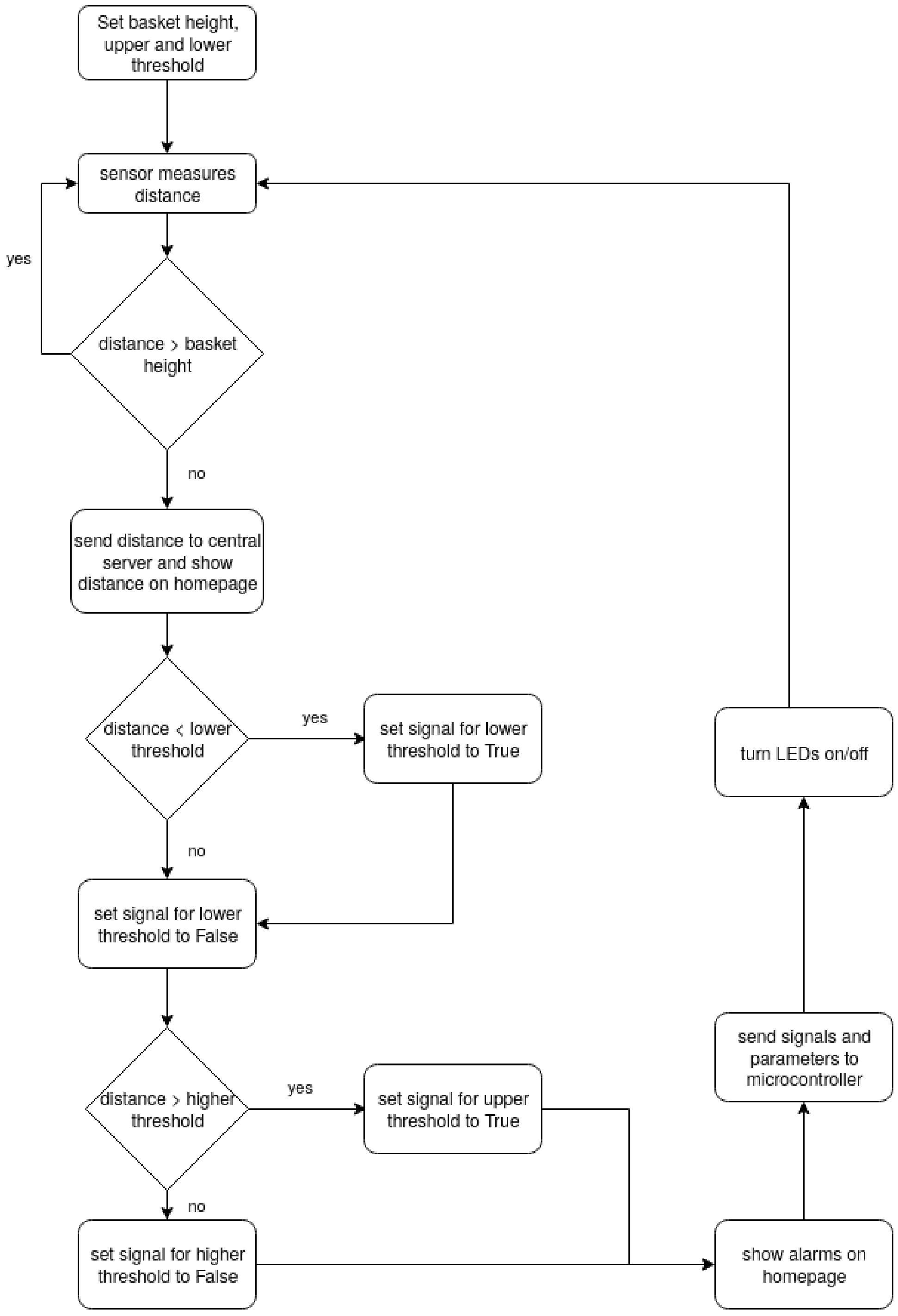

Set basket height, upper and lower threshold
sensor measures distance
yes
distance > basket height
no
send distance to central server and show distance on homepage
distance < lower threshold
yes
set signal for lower threshold to True
no
set signal for lower threshold to False
distance > higher threshold
yes
set signal for upper threshold to True
no
set signal for higher threshold to False
show alarms on homepage
send signals and parameters to microcontroller
turn LEDs on/off

Figure 4: flow chart of the system

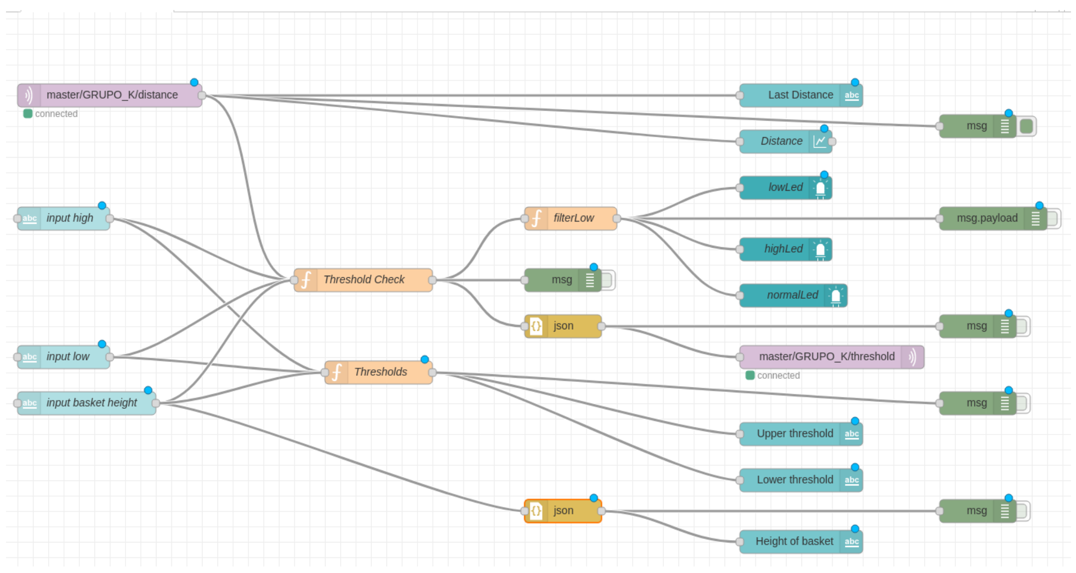
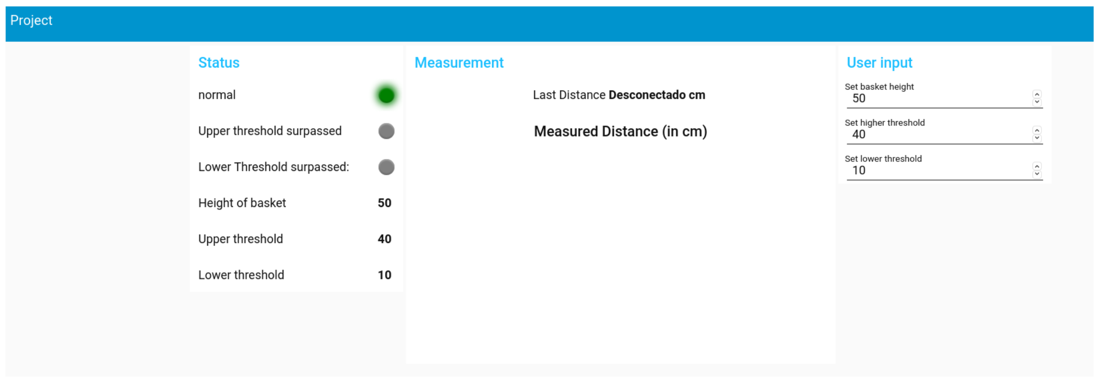


Figure 5: Node-Red: Scheme of flow


Project

Status

normal ●

Upper threshold surpassed ●

Lower Threshold surpassed: ●

Height of basket **50**

Upper threshold **40**

Lower threshold **10**

Measurement

Last Distance **Desconectado cm**

Measured Distance (in cm)

User input

Set basket height
50

Set higher threshold
40

Set lower threshold
10

Figure 6: Node-Red: Interface offline

4 Power Consumption

During the last class we measured the power consumption of the system. The measurement happenend over a duration of 8.3 seconds and the results can be seen in figure 7.

As seen in the graph the startup probably lasted around 3 seconds. In the beginning of it there are several spikes in power consumption. They could be the result of initial processes but we can't know for sure what is going on inside the ESP module. During 2.5 and 3 seconds there is a generally higher power consumption without big spikes. We think this is the period of time during which the ESP tries to establish a Wifi connection. Due to constant package traffic during this time a higher consumption could result.

Afterwards there are a lot of spikes in power consumption which probably happened because the microcontroller periodically sends and receives data from the sensor which is sent together with other packages via Wifi. Another reason could also be if the connected LEDs are turned on or off but this is hard to tell given only this sample.

If one wants to minimize the power consumption of the system, one could lower the frequency of mesurements performed by the sensor and therefore also the frequency of data exchange between the MQTT broker and the microcontroller. Also if the LEDs connected to the microcontroller aren't needed (because they are not visible or not important at the basket) they should be removed, so that the system becomes more efficient.

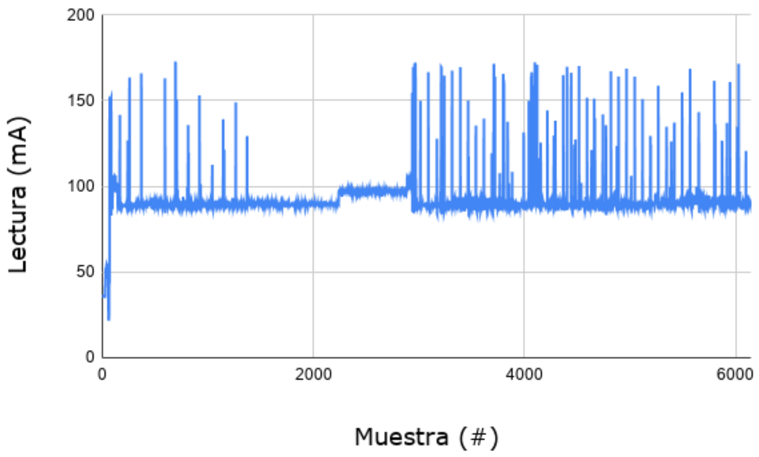


Figure 7: graph of power consumption (until 8 seconds after startup)

5 User manual and setup

The system can be set up in a few simple steps. The main prerequisite for a successful setup is a stable Wifi over which the MQTT messages can be sent to and received from the MQTT broker. The login has to be configured in the code for the microcontroller as well as for the Node-Red homepage.

Afterwards the microcontroller has to be connected to a power source and the ultrasonic sensor has to be fixed to the top of the container in which it's used pointing downwards towards the liquid stored in the basket. The rest of the hardware should be placed outside of the basket to prevent possible damage from the liquid and to make the warning LEDs easily visible.

As soon as the microcontroller is connected to a power supply it tries to connect to the given Wifi. If this is successful it starts measuring the height and sending the data to the MQTT broker. If Node-Red is running on the central server hosting our homepage it will start communicating with the MQTT broker as soon as the receiving and sending node are configured using the correct credentials.

Once this initial setup is done the system should be running and the thresholds and the basket height can be set or dynamically changed to use the full spectrum of features.

In case the microcontroller disconnects from the MQTT broker unexpectedly the message `Desconectado` is shown instead of the last measured distance. In this case the established connection should be checked as well as the microcontroller at the basket because it may have lost power. In case the error is more complex the Arduino IDE can be used as the system does extensive logging.

6 Conclusions

Through this project we were able to learn certain difficulties and also advantages of the ... There was new insight in how JSON BLA BLA In the future it would be interesting to study the capability of interconnecting with other devices and creating a more broad network of devices. Obviously it makes sense to work with: ...