

# IoT System Project - Irrigation Sprinkler

Tim Reiprich, Tegshigzugder Otgonbayar

June 23, 2020

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>General description</b>	<b>2</b>
<b>3</b>	<b>Hardware Scheme</b>	<b>2</b>
<b>4</b>	<b>Implementation</b>	<b>3</b>
4.1	project.ino . . . . .	4
4.2	mqtt.py . . . . .	4
4.3	main.py . . . . .	4
<b>5</b>	<b>Google Cloud Platform</b>	<b>6</b>

## 1 Introduction

In the summer season one of the crucial tasks for people in agricultural field is to provide sufficient water to their crops but also not to overwater. Therefore we have chosen to develop an IoT System Project for managing and regulating irrigation sprinklers automatically and smartly. The use case can be also extended to people for residential and industrial usage.

## 2 General description

In our project we have three main sensors: light sensor, temperature sensor, and distance sensor. Each sensor has an independent function. Furthermore there is a possibility to combine these sensors for more complex usage. In the following the independent functions of the sensors are described:

- Light sensor: to sense if it is day or night, so that we can decide when to activate the sprinklers
- Temperature sensor: to sense how warm the temperature is, so that the sprinklers are distributing water for a longer or shorter time
- Distance sensor: to sense if an object such as a door is open or closed, so that the sprinklers turn off e.g. if a person enters the field

For the development of this project we have mainly used the applications and tools introduced in the course. Due to the current pandemic we have opted to build the circuit in the online simulator Tinkercad instead of a physical circuit. This also enabled us easily to work together. The source code is to be found in our github repository [/urlhttps://github.com/ajpqa/iot2](https://github.com/ajpqa/iot2). The hardware scheme is to be found under <https://www.tinkercad.com/things/9b2RV6Rs0I2-projectfinal/editel>.

## 3 Hardware Scheme

In this part we will describe our the scheme of our project, which is to be seen in figure 1. It consists of an Arduino Uno which functions as a micro-controller and collects the data of the environment to control the sprinklers. The sensors to collect data are a temperature sensor, a distance sensor and a light sensor. We measure the temperature because the plants need more water if it's warmer and less if the temperature isn't that high. Furthermore the light sensor is used to determine a good day time to water the plants. Preferably it should always happen in the evening or night as there is a higher chance people work there during the day and plants can get "burned" if there is strong sun light which gets amplified by water drops on the plants. And lastly we use a distance sensor to determine if the door to the greenhouse is open or closed. If it is open there

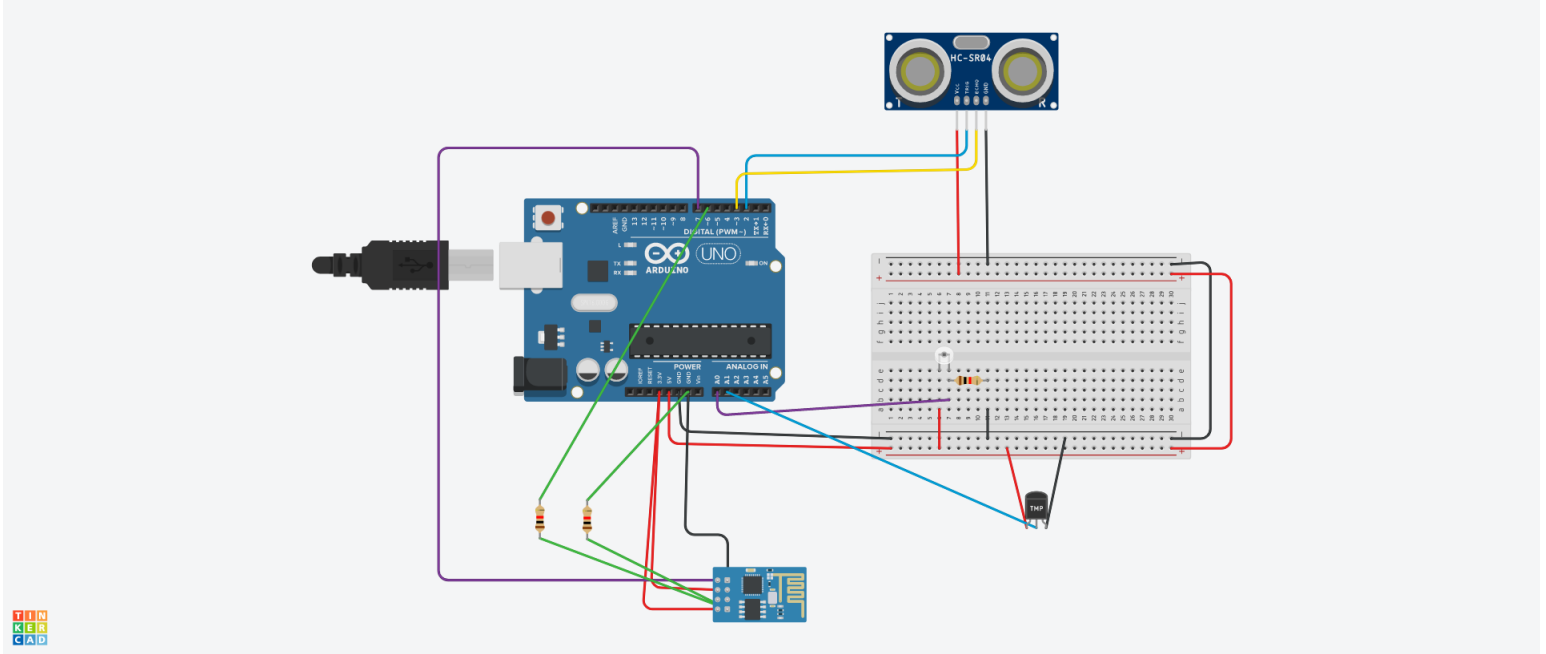


Figure 1: Circuit that was built during the project

are workers inside which means that the sprinklers have to be turned off. Only after the door is closed the sprinkler can be turned on again.

The last component of the hardware scheme is the ESP8266 Wifi module which is used to send the measured data to a user interface and Google Cloud using the MQTT protocol. Sadly the Tinkercad circuit simulator doesn't support the Arduino library `PubSubClient.h` which leads to us not being able to test our configuration properly. In theory the provided code should work in a real environment but sadly the limitations of the simulator are too narrow. So we used a python script to simulate sending the data to the MQTT broker, so that our circuit, user interface and evaluation of the data in Google Cloud works but the transmission of the data has to be simulated by us.

## 4 Implementation

The implementation of the project was written in the programming languages C++ and Python. The project consists of the following main parts:

- `project.ino` - program for the micro controller
- `mqtt.py` - the mqtt simulation and virtual environment
- `main.py` - the user interface for the local view and control of sensors and sprinklers

## 4.1 project.ino

This file contains the code controlling the Arduino. It's main purpose is to collect the sensor data, convert in humanly standardized measurements and send it via MQTT to a broker. The file can be seen in the Tinkercad project <https://www.tinkercad.com/things/9b2RV6Rs0I2>.

It consists of multiple functions. Two own helper functions were written. `sendData` handles sending commands to the ESP8266 Wifi module and `readUltraSonicDistance` handles measuring the distance as it requires more effort than the other sensors. The two basic functions are `setup()` and `loop()`. `setup()` initializes all variables and sets up the connection to the Wifi while `loop()` periodically gets the sensors' measurements, prints them to the serial monitor and finally converts the data into a JSON and publishes it via MQTT. Sadly due to restrictions of the Tinkercad simulator the last part can't be used in the simulator as the MQTT library is not available there. Additionally there is a library handling the Wifi connection using the ESP8266 which is also not supported. So we put all the code regarding MQTT and the Wifi library in the file as comments. It should work in a real environment but we couldn't test it as the libraries are just not available. Furthermore we created a python file `mqtt.py` as a work-around which simulates the publishing of the data (see next chapter).

## 4.2 mqtt.py

This is an additional file which would not be needed if we could have used a real circuit but due to the special circumstances we use it to simulate publishing the measured data via MQTT. It simply contains three functions each simulating a sensor and publishes the "measured" data to the Google Cloud Platform and the hivemind broker where the user interface gets its data from.

## 4.3 main.py

The User Interface is to be found here. It was mainly created with the widgets of tkinter and tkkinter. It consists of the following three pages; control, value, and summarize.

**Control page** Here is the main control and view of the system. In the "Sensors control" box we can start or stop the receiving of the values of the sensors. In this case depending on a boolean value the received values of the subscription is either inserted or not. In the "Current" values box the most recent received values are shown. The main feature of the distance sensor is to stop the sprinklers in the case that a person or object is moving to the field. In "Automatic Stop" we can set a distance threshold when the trigger object is too close. For example if the object is a door and the door is opened, the sprinkler stops for a period of time until the door is closed again. Here the status is either "Under

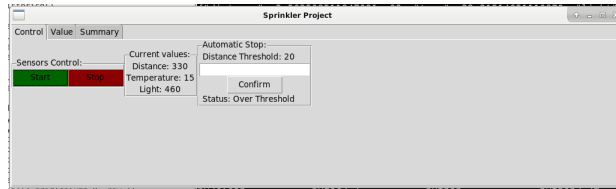


Figure 2: Control Page

#Num	Distance	Temperature	Light	Date
1	6.088419396892632e-13	36.2432005341180066	360	Sun Jun 21 14:11:01 2020
2	3.7210723976820985e-13	35.365601899630796	370	Sun Jun 21 14:11:02 2020
3	2.637866557591424e-13	38.55109365630865	380	Sun Jun 21 14:11:03 2020
4	2.6585029984894484e-14	38.40033420682867	390	Sun Jun 21 14:11:04 2020
5	1.8020024847732197e-14	37.35422685424031	400	Sun Jun 21 14:11:05 2020
6	1.3931247573968865e-14	40.660677048337305	410	Sun Jun 21 14:11:06 2020
7	5.324226293611861e-15	11.674534648251075	420	Sun Jun 21 14:11:07 2020
8	4.397988201748444e-15	12.456716992880722	430	Sun Jun 21 14:11:08 2020
9	2.017240598297795e-15	13.095503175001651	440	Sun Jun 21 14:11:09 2020
10	1.6980197904390926e-15	14.11265369421484	450	Sun Jun 21 14:11:10 2020

Figure 3: Value page

Threshold” or ”Over Threshold”. By clicking on the ”Confirm” button we can set a new distance threshold.

**Value page** While in the ”Control” page we can only see the most recent values, in the ”Value” page all of the values are registered. This can be useful for further analyzing and evaluating based on specific needs of the user.

**Summarize page** The summarize page functions to evaluate the values. With the ”Summarize” button the average ”temperature” and ”light” values are calculated.



Figure 4: Summarize page

## 5 Google Cloud Platform

To be able to store the measured data for later analysis we decided to use the Google Cloud Platform as it provides enough storage as well as multiple tools to analyse the stored data.

We used a similar approach like during the classes. We created a new registry and device to receive the data via MQTT using the Pub/Sub tool provided by the Google Cloud Platform. As seen in figure 5 the data is received successfully.

To permanently store the received data we used a Cloud Function to store the data in a SQL table which we created using BigQuery. (see figure 6) From there it can be easily extracted to either look at certain data points or analyze it using further tools.

Tabelle filtern						
Nachrichtentext	JSON-Schlüssel im Text		Attribut.deviceId	Attribut.deviceNumId	Attribut.deviceR	Bestätigen ↑
{"temp": 27.1589407...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 28.4132589...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 27.3339364...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 26.7917826...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 29.1981363...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 27.6394386...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 25.0393740...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 23.8636512...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 23.831278880533315, "distance": 330, "light": 20}	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ^
	light					
{"temp": 24.0388448...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 24.9337028...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 24.0471610...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 29.8930228...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 28.6733369...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 26.4382285...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 27.9157713...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 24.5554547...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼
{"temp": 16.5751452...	temp	distance	my-device	2876321708708126	data-project	Frist überschritt... ▼

Figure 5: received messages via the Pub/Sub tool



Google Cloud Platform

My Project 65608

Produkte und

BigQuery

FEATURES UND INFORMATIONEN

TASTENKOMBINATION

Abfrageverlauf

Gespeicherte Abfragen

Jobverlauf

Übertragungen

Geplante Abfragen

Reservierungen

BI Engine

Ressourcen

Tabellen und Datasets suchen

arctic-defender-275118

bici

final\_project

myTable

midataset

Abfrageeditor

1

SELECT \* FROM `arctic-defender-275118.final\_project.myTable` LIMIT 1000

Ausführen

Abfrage speichern

Ansicht speichern

Abfrage planen

Mehr

Abfrageergebnisse

ERGEBNISSE SPEICHERN

DATEN AUSWERTEN

Abfrage abgeschlossen (0,8 s verstrichen, 0 B verarbeitet)

Jobinformationen

Ergebnisse

JSON

Ausführungsdetails

Zeile	temp	distance	light	now
1	31.209561757247627	330.0	390	2020-06-23T14:30:57.375135
2	17.914346370899754	31.435798701208164	280	2020-06-23T14:31:24.427243
3	30.692421010189737	5.440638098574977	470	2020-06-23T14:31:05.387931
4	17.756434813145233	15.289528623535995	270	2020-06-23T14:31:25.429506
5	15.978433803888693	330.0	320	2020-06-23T14:31:20.418354
6	29.92945587300256	330.0	230	2020-06-23T14:30:41.342346
7	31.242984900984084	0.0021574113584388647	370	2020-06-23T14:30:55.370819
8	33.699290487124614	1.4675964933256391	450	2020-06-23T14:31:07.392183
9	27.688397516020036	0.055896577850367356	350	2020-06-23T14:30:53.366627
10	13.487215447718256	330.0	340	2020-06-23T14:31:18.414658

Figure 6: Stored measurements in a BigQuery table