Embedded systems 2

Report Cosy Cafeteria

Robin Van de Poel Arthur Van den Storme
Tobias Cromheecke Thomas Feys

May 6, 2020

Contents

1 Intro

To improve the experience of the students in the cafeteria, an embedded system will be developed to gather a variety of data. This data will then be display to the students through an online medium such as a website. The main focus will be to gather information about the amount of people that are in the cafeteria. There are two main goals. The first goal is to display the estimated waiting time to get a meal to the students. Secondly the amount of available seats will be tracked and displayed to the students. Next to this some additional information will be gathered. The temperature, air-quality and sound level will be monitored to give the students a general idea about the conditions in the cafeteria. This way the students can gauge if the cafeteria is suitable to study in at a certain moment. An overview of the envisioned system is displayed in figure 1. All the developed hardware and software is displayed at: `https://github.com/thomasf10/cosy_cafeteria`.

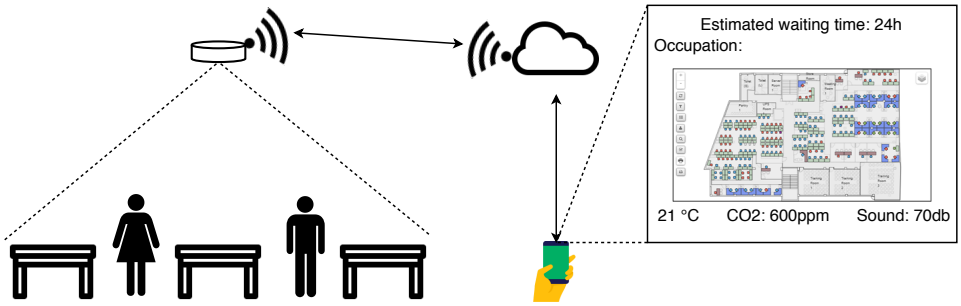


Figure 1: Situation sketch

2 Specifications

2.1 Functional specifications

- Monitor the available seats
- Monitor the queue to get a meal
- Display the data to the students
- Energy provision for 1 semester (4-5 months)
- Wireless coverage over the entire campus
- Measure new data every 10 minutes
- Send the data to the online medium every hour
- If the data changes with a large amount (needs to be specified) update the online medium sooner

2.2 Other technical specifications

- Enclosure: 10 x 10 cm
- Easy deployment: the node must be easy to attach and detach from the sealing
- Must be easy to recharge (Wireless or via USB)
- Light weight: 300 g (so it wont detach from the sealing)

2.3 Non-technical specifications

- Cost: €??
- The data should be easy accessible to the students

3 System architecture

An overview of the full system is displayed in figure 2. The developed system uses three sensors:

- AMG8833: IR grid sensor
- CSS811: air quality sensors
- Sparkfun sound detector: microphone

By incorporating these sensors in to the system, a variety of sensor data is gathered, namely: ambient temperature [°C], CO_2-level [ppm], TVOC-level (total volatile organic compounds) [ppb] and sound level [db]. Next to this the AMG8833 captures the temperature af a two dimensional area in 8 by 8 pixels.

The 'brain' of the system is an ESP32, this microcontroller has an integrated Wi-Fi module, which will be used to transmit the data. The gathered data is sent to a server, which stores the received data in a database.

Alongside the reception of data from the ESP32, the server hosts a website which is used to display the data to the students. The website uses the pixel data to generate a heatmap of the cafeteria, this allows the students to see how many people are present in the cafeteria. Unfortunately one module cannot cover the entire cafeteria, to generate the heatmap. In order to do this, several modules have to be deployed in the cafeteria. In this project only one module will be constructed as a proof of concept. Next to the heatmap, the other sensor data is displayed on the website this includes: ambient temperature, CO_2-level, TVOC-level and the sound-level. Based on

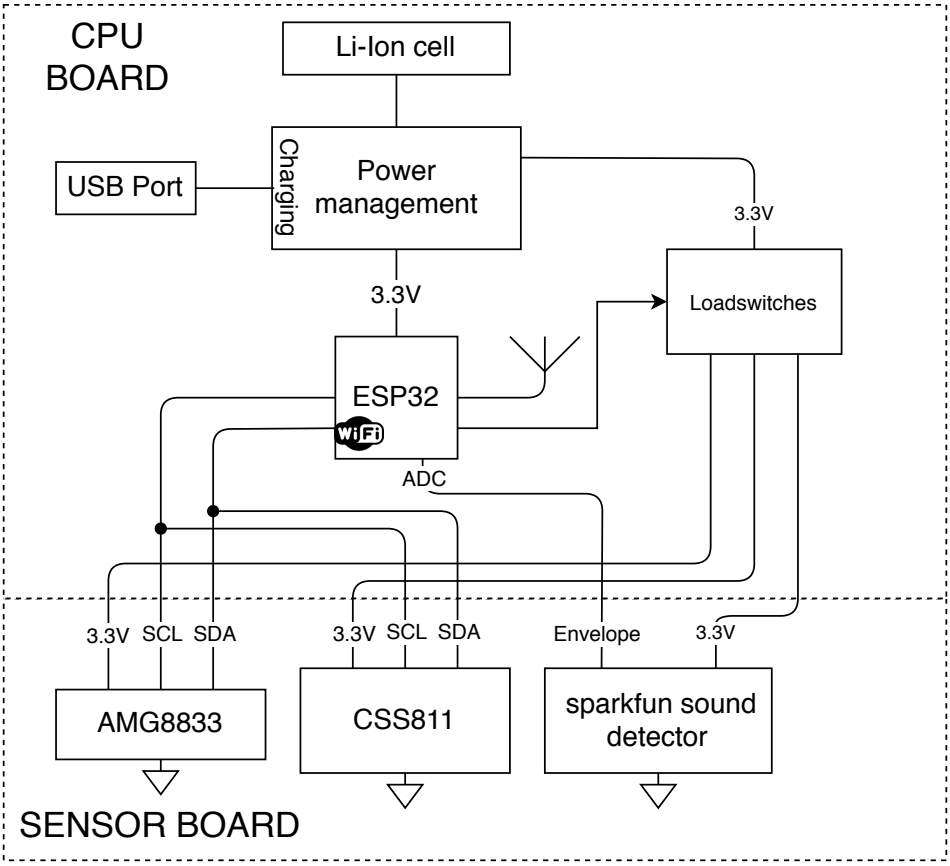


Figure 2: System architecture

these parameters the students can gauge if the cafeteria is currently suitable as a study-location.

4 Power management design

This section explains the different trade-off's which were made in designing the power management part of the system. The first part in designing the power management is choosing a way to supply the circuit. In our case, the device needs to be deployed at the ceiling of a cafeteria. You can't just tap off the electricity cable at any point here, so our device will need a battery to supply the power. Our first aim was to supply the device as long as one semester. Another aspect was the ability to recharge the battery for environmental reasons. This excludes standard cell batteries that have a much lower energy density and are not rechargeable. Our choice was quickly made for the type of battery, namely a Lithium-Ion (Li-Ion) battery which will be discussed in section 4.1.

4.1 Lithium-Ion battery

Lithium-Ion batteries are now a commonly used battery in consumer electronics. Li-Ion batteries is the most commonly used battery in phones, MP3 players and other portable equipment. The voltage of a single Li-Ion battery cell is usually fixed at 3.6 V or 3.7 V. We call this the nominal voltage. However, the actual voltage during use can vary from 2.4 V to 4.2 V. When the voltage drops below 2.4 V, the cell is discharged too deeply and damaged beyond repair. Therefore, a safety voltage of 3 V is often maintained. If the cell is discharged to this voltage, nothing is wrong. Charging a Li-Ion battery usually takes 3 hours. Don't be blinded by fast chargers for these batteries, practical tests have shown that it takes 3 hours until a battery is fully charged [1]. A few benefits of a Li-Ion battery are:

- Very high energy density
- Minor self-discharge
- No memory effect
- High power, long service life

Some disadvantages are:

- Relatively expensive to purchase
- Dangerous if used incorrectly, can explode/combust
- Need a protection circuit to not drop below the 2.4 V threshold value
- Charging and discharging process needed

The most common Li-Ion battery is the 18650 cylindrical cell which is 18mm wide, 65mm long and the '0' stands for its cylindrical shape. Another common type of Lithium-Ion battery is the Lithium-Ion Polymer (Li-Po) battery. Li-Po's are preferred over standard Li-Ion batteries in application where weight reduction is important, fast charge opportunity is needed and which need a very high current during discharge time. Li-Po's are more used for drones (where lightweight is a requirement) and electric cars (fast charging requirement). Li-Po's are also less likely to combust or explode and are more expensive. Therefor a Li-Po would be overkill in our application [**?** 2**?**].

4.1.1 Selected battery

The reason we chose a single cell Li-Ion battery was:

- Voltage range: our microcontroller needs 3.3 V supply voltage, which is only a low dropout voltage of 0.4 V in comparison with the 3.7 V nominal voltage of a single cell Li-Ion battery. This results in less power dissipation in the voltage regulator.

- Rechargeable: for environmental aspects
- High capacity: around 3000 mAh, to last a semester

The specifications of the chosen NCR18650B are:

- Size/package: 18650
- Rated capacity: 3200 mAh
- Nominal voltage: 3.6 V
- Maximum voltage: 4.2 V
- Maximum charge current: 1625 mA
- Maximum discharge current: 6400 mA[1]
- Maximum discharge voltage: 2.5 V
- Charge time: 4 hours
- No protection circuit

To charge the Li-Ion we had to include a charge circuit in our design. We found a high frequently used breakout board to charge a single cell Li-Ion battery via USB with battery protection here. Such a charge circuit would be perfect for our application. Due the reason we couldn't use a breakout board we had to integrate the components on one PCB. Unfortunately the components which were used on the breakout board were not available on the websites we're using to order components like Mouser, Digi-Key, Farnell, etc. After some research and a recommendation of a fellow student who had experience with battery charging applications in his master thesis, he advised to go for Texas Instruments IC's for charging Li-Ion batteries. The charger IC didn't have a protection circuit on it, so we also needed a circuit for this. Eventually we also went for a protection IC and voltage regulator of Texas Instruments. To protect the circuit for reverse polarity of the battery we also included an self-designed protection circuit which was first evaluated in LT Spice. All named components will be discussed in the following sections.

4.2 Reverse polarity protection

Reverse polarity protection is a nice feature to have te protect a circuit against polarity mismatches of the battery. For the design of this circuit we based us on a design found on the internet here. Before implementing this circuit in our design we first evaluated it's correct functioning using LT Spice in section 4.2.1.

[1]Maximum discharge rate is 2C. This means discharging the battery capacity in 0.5 hours $\frac{3200mAh}{0.5h} = 6400mA$.

4.2.1 Simulation

Reverse polarity protection is simply done by including a P type MOSFET as shown in figure 3. Since the maximum voltage of the battery is limited to 4.2 V, there is no need for high drain-to-source voltage V_{DS} requirements for the MOSFET. A suitable MOSFET was found [3]. Normally there should be a zener diode placed between gate and source to protect the maximum gate-to-source voltage V_{GS}. Since the selected MOSFET's $V_{DS} = V_{GS} = 8$ V there is no need for a zener diode to clamp the voltage.

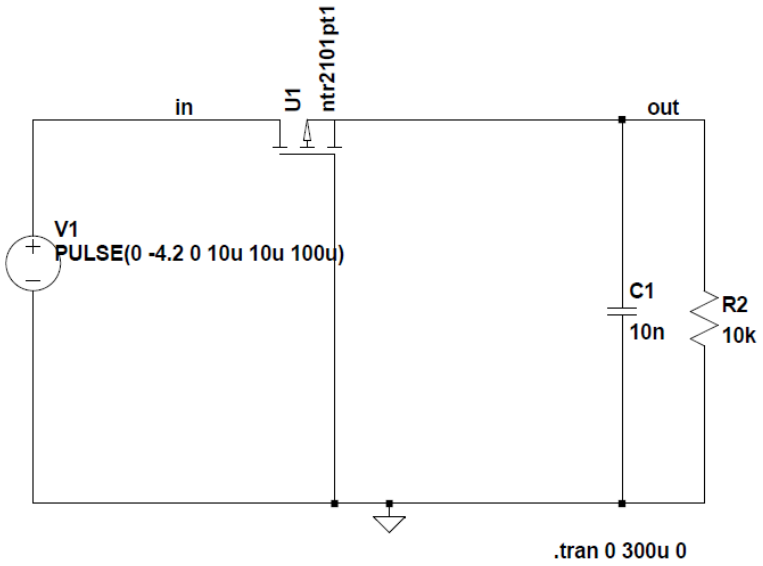


Figure 3: Simulation reverse polarity protection circuit

Figure 4 shows the transient response when applying a negative pulse at the input. As can be seen in the figure, the negative pulse doesn't come through the rest of the circuit. The voltage at the output is 0 V.

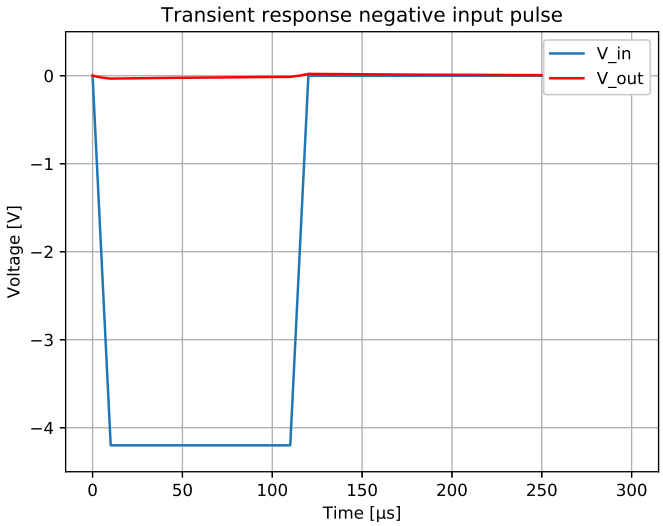


Figure 4: Transient response negative input pulse

When applying a positive pulse at the input, it can be seen in figure 5 the output follows the input. We can evaluate the circuit works properly to protect against reverse polarity protection. Now it can be implemented in our design discussed in section 4.2.2.

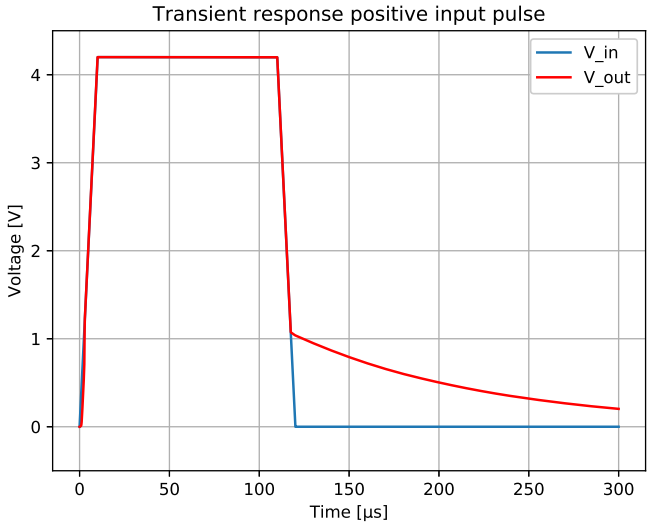


Figure 5: Transient response positive input pulse

4.2.2 Implementation

As a precaution we included two batteries in the first version of our design. This was done for debugging reasons: in case our real current consumption would be too high after trying every possibility to reduce it in software, we had an extra battery to meet our lifetime goal. It was possible to include an extra battery because the PCB dimensions were still lesser then 100 x 100 mm. JP1 is a footprint to solder a wire so we are able to measure the current using a current probe. When the battery is inserted with the correct polarity, +BATT is connected to VDD1.

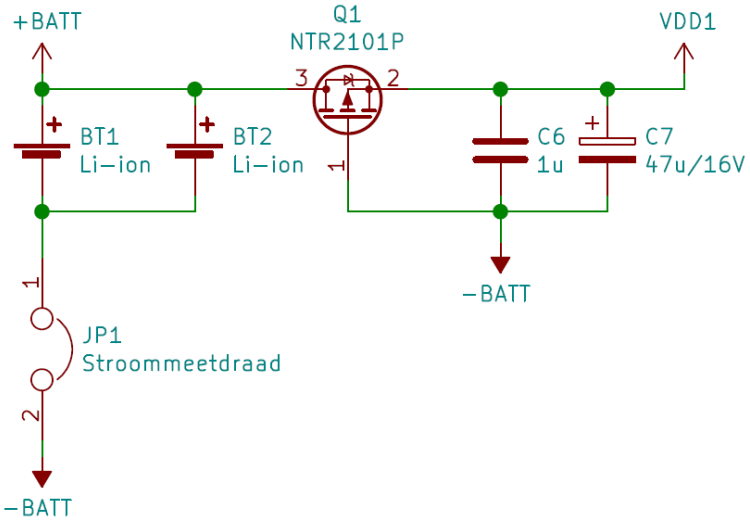


Figure 6: Reverse polarity protection circuit

4.3 Charging choice

For charging you're free to choose at which voltage and current you charge. Eventually a Li-Ion charger IC will step down the voltage to the maximum 4.2 V for a cell. For charging you could design your own AC-DC converter, we went for the ability to charge the device using USB. Here for we rely on a power adapter which transforms 240 VAC to 5 VDC, or the ability to charge our device using an USB port on a computer. Figure 7 shows the input connector micro USB type B with two decoupling capacitors C_1 and C_2. The charging voltage source is labeled as Vin.

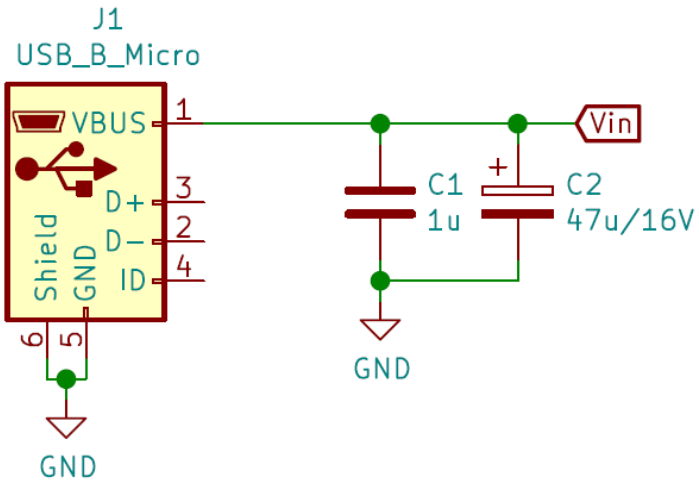


Figure 7: Micro USB type B input

4.4 Battery charging circuit

We considerd many chips from different kind of manufacturers. As previously said a TP4056 IC charger is commonly used for single cell Li-Ion batteries, but it's not available on Digi-Key, Farnell, etc. Eventually we went with the BQ24075 IC from Texas Instruments. It's a great choice because it's made to charge a single cell Li-Ion battery, and it's perfectly fit to charge using a USB port. The IC operates from either a USB port or an AC adapter and support charge currents up to 1.5 A. The input voltage range with input overvoltage protection supports unregulated adapters. The USB input current limit accuracy and start up sequence allow the BQ24075 to meet USB-IF inrush current specifications[2]. Additionally, the input dynamic power management (VIN-DPM) prevents the charger from crashing incorrectly configured USB sources. An additional specification that was decisive in the choice of charger IC was the availability of 2 pins for status indication:

1. **power good indication:** if the input voltage Vin is in specified range.
2. **charge indication:** if the charger is charging (bright up LED) and when the charge cycle is complete (quench LED)

Figure 8 shows the BQ24075 IC implemented in our design. A green LED D1 indicates the power good signal, and blue LED D2 indicates the charging stage. When no voltage is applied at the input Vin, the internal transistor Q1 is opened and Q2 is closed. Therefor VDD1 is connected to VDD2. During charge cycle both transistors Q1 and Q2 are closed allowing the battery

[2] `http://www.testusb.com/inrush_issue.htm`

to charge. The BQ24075 feature a SYSOFF input that allows the user to turn Q2 off and disconnect the battery from the OUT pin. This is useful for disconnecting the system load from the battery, factory programming where the battery is not installed or for host side impedance track fuel gauging, where the battery open circuit voltage level must be detected before the battery charges or discharges. We don't use this feature so we connect SYSOFF to GND to close Q2 for normal operating mode [4]. The other features of BQ24075 and schematic components will be discussed in sections 4.4.1 to 4.4.6.

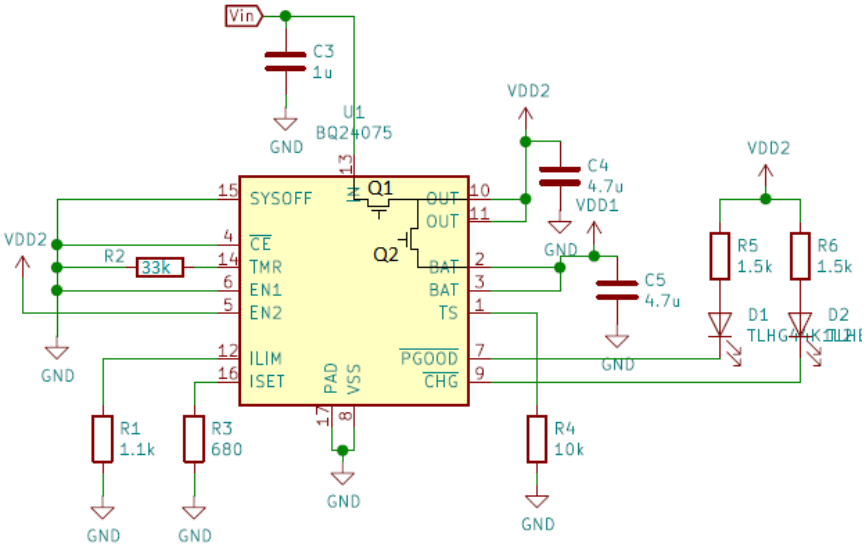


Figure 8: Typical application circuit BQ24075

4.4.1 Charging phases

Set $\overline{CE}$ low to initiate battery charging. The battery is charged in three phases:

1. Conditioning pre-charge
2. Constant current (CC) fast charge (current regulation)
3. Constant voltage (CV) tapering (voltage regulation)

Figure 9 illustrates the three charging phases. In the pre-charge phase, the battery is charged with the pre-charge current I_{PRECHG}. Once the battery voltage crosses the V_{LOWV} threshold, the battery is charged with the fast-charge current I_{CHG}. As the battery voltage reaches $V_{BAT(REG)}$, the battery is held at a constant voltage of $V_{BAT(REG)}$ and the charge current tapers off as the battery approaches full charge. When the battery current

reaches I_{TERM}, the CHG pin indicates the charging is done by going high-impedance.

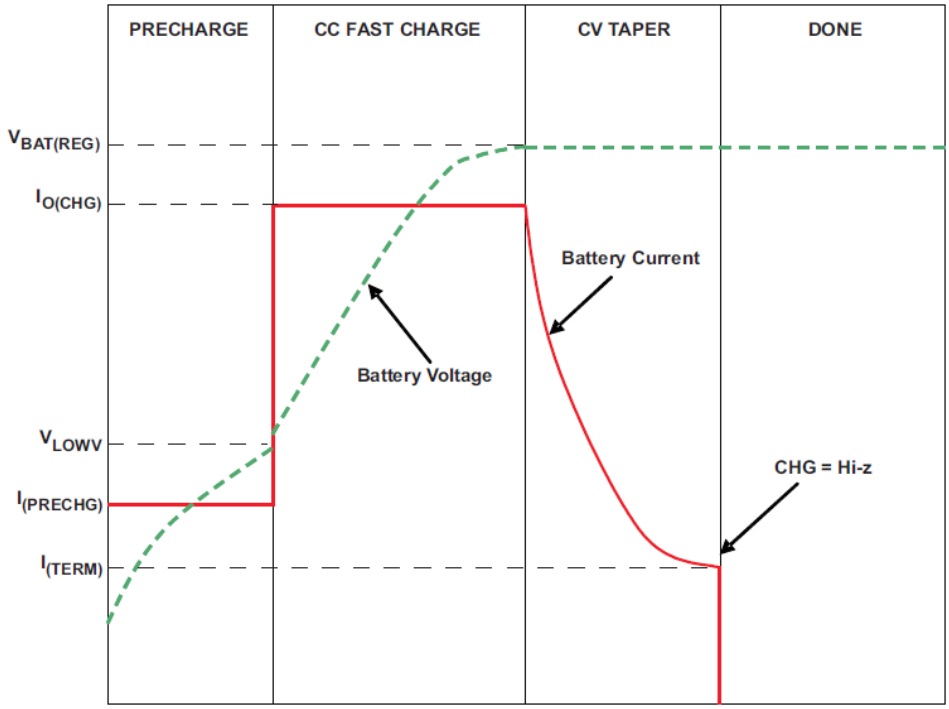


Figure 9: Charge cycle [4]

4.4.2 Set input current limit I_{LIM}

The BQ24075 IC offers a fully compliant USB charger, meaning:

1. Selectable 100 mA and 500 mA maximum input current
2. 100 mA Maximum current limit ensures compliance to USB-IF standard
3. Input-based dynamic power management (VINDPM) for protection against poor USB sources

As shown in table 1, EN1 and EN2 are used to configure the USB charge current.

Table 1: EN1/EN2 Settings [4]

EN2	EN1	MAXIMUM INPUT CURRENT INTO IN PIN
0	0	100 mA. USB100 mode
0	1	500 mA. USB500 mode
1	0	Set by an external resistor from ILIM to VSS (max. 1.5 A)
1	1	Standby (USB suspend mode)

USB1.0 specifications allow devices to draw up to 500 mA from one port. USB operates at 5V, so that means a maximum of 2.5 watts. USB 2.0 has the same power limit. But later there were add-ons to the spec for battery charging, allowing up to 1.5A (7.5W) while data transfer is going on, and up to 5A if not. Few USB 2.0 devices can deliver that much power. Also note that the Micro USB connector is only rated to carry 2.1A (10.5W). USB 3.0 increases the power limit to 900mA (4.5W). The battery charging spec from USB 2.0 can optionally be supported [5]. Since almost all USB chargers and ports are 2.0 and 3.0 nowadays, we can step up the current above 500 mA. The value of the input current limit I_{LIM} is set by the resistor connected from the ILIM pin (pin 12) to VSS, and is given by [4]

$$R_{ILIM} = \frac{K_{ILIM}}{I_{LIM}}\,, \tag{1}$$

with

$$K_{ILIM} = 1610A\Omega\,. \tag{2}$$

The valid resistor range is 1.1 kΩ to 8 kΩ. A value of 1.1 kΩ results in an input current limit

$$I_{LIM} = \frac{1610A\Omega}{1.1k\Omega} = 1.46A\,. \tag{3}$$

The maximum input current limit 1.5 A is reached. **Caution:** make sure this current doesn't exceed the value of the maximum charge current of the battery, which is in our case within the permitted limit of 1625 mA (section 4.1.1).

4.4.3 Fast charge safety timer (TMR)

TMR controls the pre-charge and fast-charge safety timers. Leave TMR open to set to default safety timers. Connect to VSS to disable safety timers. Connect a 18 kΩ to 72 kΩ resistor between TMR and VSS to program the timers a desired length. Reset the timers by toggling the CE pin, or by toggling EN1, EN2 pin to put the device in and out of USB suspend mode (EN1 = 1, EN2 = 1). The charge time of the battery is 4 hours (section 4.1.1). We will calculate the TMR resistor R_{TMR} to meet the desired charge time of 4 hours, using equation [4]

$$R_{TMR} = \frac{t_{MAXCHG}}{10 \cdot K_{TMR}}\,, \tag{4}$$

with

$$K_{TMR} = 48\,s/k\Omega\,, \tag{5}$$

$$t_{MAXCHG} = 4hr\,. \tag{6}$$

$$R_{TMR} = \frac{t_{MAXCHG}}{10 \cdot K_{TMR}} = \frac{4\,hr \cdot 3600\,s/hr}{10 \cdot 48\,s/k\Omega} = 30\,k\Omega \tag{7}$$

Select a resistor value of 33 kΩ, so it's E6-E12 compliant an to extend it charge time a little bit. Connect this resistor between TMR (pin 2) and VSS.

4.4.4 Set fast charge current I_{CHG}

The value of fast-charge current I_{CHG} is set by the resistor connected from the ISET pin to VSS, and is given by [4]

$$R_{ISET} = \frac{K_{ISET}}{I_{CHG}}\,, \tag{8}$$

with

$$K_{ISET} = 890A\Omega\,. \tag{9}$$

The charge current limit is adjustable up to 1.5 A. The valid resistor range is 590 Ω to 8.9 kΩ. If I_{CHG} is programmed as greater than the input current limit I_{LIM}, the battery will charge at the rate of I_{LIM} instead of I_{CHG}. In this case, the charger timers will be proportionately slowed down. The maximum charge current for the Panasonic NCR18650B battery is 1625 mA (section 4.1.1). Set charge current to 1.3 A to have some margin for the battery lifetime and calculate the resistor value

$$R_{ISET} = \frac{890A\Omega}{1.3A} = 684.62\Omega \approx 680\Omega\,. \tag{10}$$

Select the closest standard value (E6,E12,...), which for this case is 680 Ω. Connect this resistor between ISET (pin 16) and VSS.

4.4.5 TS function

The TS-pin is the external NTC thermistor input. Connect the TS input to the NTC thermistor in the battery pack. TS monitors a 10 kΩ NTC thermistor. For applications that do not use the TS function, connect a 10 kΩ fixed resistor from TS to VSS to maintain a valid voltage level on TS.

4.4.6 Selecting IN-, OUT-, and BAT-pin capacitors

In most applications, all that is needed is a high-frequency decoupling capacitor (ceramic) on the power pin, input, output and battery pins. Using the values shown on figure 8 (these are taken from the typical application circuit in [4]) is recommended. After evaluation of these voltage signals with real system operational conditions, one can determine if capacitance values

can be adjusted toward the minimum recommended values (DC load application) or higher values for fast high amplitude pulsed load applications. Note if designed high input voltage sources (bad/wrong adaptors), the capacitor needs to be rated appropriately. Ceramic capacitors are tested to 2x their rated values so a 16-V capacitor may be adequate for a 30-V transient (verify tested rating with capacitor manufacturer).

4.5 Battery protection circuit

Because the battery we selected doesn't have an integrated protection circuit, we also needed to consider a battery protection circuit in our design. As discussed in section 4.1 it could lead to battery defect when the voltage of the battery drops below 2.4 V threshold value. A Li-Ion is also likely to explode/combust when charging too high in voltage or current. Therefor a protection circuit is always needed. We went for the BQ29700 which is a battery cell protection IC that provides an accurate monitor and trigger threshold for overcurrent protection during high discharge/charge current operation or battery overcharge conditions. The BQ29700 provides the protection functions for Li-Ion/Li-Po cells, and monitors across the external power FETs for protection due to high charge or discharge currents. Figure 10 shows a typical application circuit of the BQ29700 IC. In normal mode FETs Charge FET (CHG) and Discharge FET (DSG) are closed, however in one of the following conditions the FETs will open to stop the current flow from/to the battery:

1. **Overcharge detection (OVP):** When the cell exceeds the overcharge detection threshold voltage V_{OVP} during charge, the safety circuit will interrupt the flow of current into the cell.When the overcharge voltage is exceeded, a delay of up to t_{OVP} will occur before the FETs open the circuit.

2. **Over-discharge detection (UVP):** When the cell exceeds the overdischarge detection threshold voltage V_{UVP} during discharge, the safety circuit will interrupt the flow of current out of the cell.When the overdischarge voltage is reached, a delay t_{UVP} will occur before the FETs open the circuit.

3. **Charge overcurrent detection (OCC):** When the pack output current the charge overcurrent detection threshold voltage V_{OCC} during charge, the safety circuit will interrupt the flow of current out of the pack. When the charge overcurrent threshold voltage is exceeded, a delay of up to t_{OCC} will occur before the FETs open the circuit.

4. **Discharge overcurrent detection (OCD):** When the pack output current the discharge overcurrent detection threshold voltage V_{OCD} during discharge, the safety circuit will interrupt the flow of current

out of the pack. When the discharge overcurrent threshold voltage is exceeded, a delay of up to t_{OCD} will occur before the FETs open the circuit.

5. **Load short-circuit detection (SCD):** Similar to overcurrent threshold values, except a short circuit will result in a much higher current, thus a higher sense voltage. The short-circuit detection threshold voltage is indicated as V_{SCD} with delay t_{SCD}.

Specifications of the BQ29700 IC are [6]:

- $V_{OVP} = 4.275$ V
- $t_{OVP} = 1.25$ s
- $V_{UVP} = 2.8$ V
- $t_{UVP} = 144$ ms
- $V_{OCC} = -0.1$ V
- $t_{OCC} = 8$ ms
- $V_{OCD} = 0.1$ V
- $t_{OCD} = 20$ ms
- $V_{SCD} = 0.5$ V
- $t_{SCD} = 250\ \mu s$

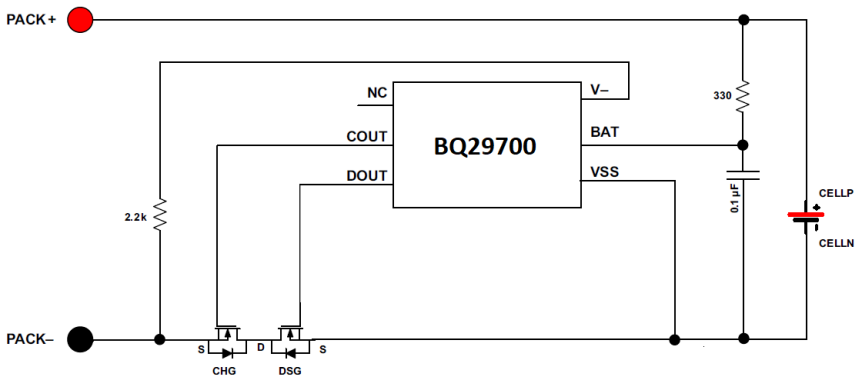


Figure 10: Typical application circuit BQ29700 [6]

4.5.1 Dimensioning

Figure 11 shows the implemented BQ29700 IC in our design. An RC filter is required on the BAT-pin for noise, and enables the device to operate during sharp negative transients. The 330 Ω resistor also limits the current during a reverse connection on the system. TI recommends placing a high impedance 5 MΩ across the gate source of each external FET to deplete any charge on the gate-source capacitance. The voltage sense node V_- is a sense node used for measuring several fault detection conditions, such as overcurrent charging or overcurrent discharging configured as Vds sensing for protection. This input, in conjunction with VSS, forms the differential measurement for the stated fault detection conditions. A 2.2 kΩ resistor is connected between this input pin and Pack– terminal of the system in the application.

FET Selection: These should be MOSFETs who operate at relativly low voltages (2.8-4.2V battery cell) and that can handle a current of 1.3 A for charging (section 4.4.4). Because the current is measured by the voltage drop across the FETs, it's R_{DSON} value is also an important value by choosing the right MOSFET. We chose FDS9926A, as it's a dual N-channel MOSFET IC. So both charge and discharge MOSFETs are integrated in one IC. Each FET's R_{DSON} = 35 mΩ at Tj = 25 °C and Vgs = 3.7 V (nominal battery voltage) [7]. Because both discharge and charge overcurrent detection levels are the same (100 mV), the discharge and charge current are limited to approximately $\frac{100mV}{2 \cdot 35m\Omega}$ = 1.43 A [6, 7].

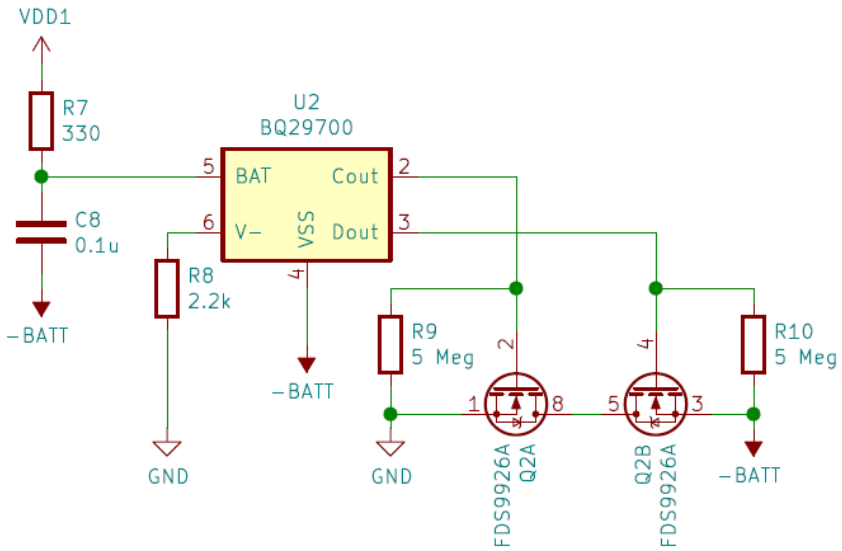


Figure 11: BQ29700 applied schematic

4.6 Voltage regulator

A few important points to consider when selecting the voltage regulator are:

1. Low quiescent current[3] I_Q to reduce power losses for extended battery lifetime.

2. Low dropout voltage[4] V_{DO} due to the small difference between the 3.7 V nominal voltage of the battery and 3.3 V supply voltage for the ESP32 microcontroller.

3. At least 500 mA maximum output current $I_{OUT,MAX}$ needed for ESP32 spikes [8].

In low power applications a linear regulator is preferred over a switching regulator due the fact switching converters are inefficient at very low loads, and power dissipation in the linear regulator is also low when combined with a low dropout voltage ($P^{\downarrow\downarrow} = U_{DO}^{\downarrow} \cdot I^{\downarrow}$). Linear regulators also have much lower quiescent currents in comparison to switching regulators [9, 10]

4.6.1 Selected regulator

As shown in figure 12 we chose TLV75533 from Texas Instruments, with the following parameters:

- $I_Q = 25\ \mu$A (Typical)
- V_{DO} = 150-238 mV (@ $I_{OUT} = 500$ mA and $V_{OUT} = 3.3$ V)
- $I_{OUT,MAX} = 500$ mA

This device features an internal soft-start to lower inrush current, thus providing a controlled voltage to the load and minimizing the input voltage drop during start up. When shutdown, the device actively pulls down the output to quickly discharge the outputs and ensure a known start-up state. For protection reasons the TLV75533 has an integrated thermal shutdown, current limit, and undervoltage lockout (UVLO).

Additionally, the TLV75533 has an enable functionality to minimize standby power. Unfortunately we can't use this function because the LDO supplies the microcontroller (these control signals to drive the enable pin should originate from the microcontroller). Therefor the enable pin is connected to the input pin to enable when an input voltage is supplied.

[3] Current which is flowing though the converter when no load is applied.

[4] The Dropout Voltage of a regulator is the amount of voltage that a regulator needs to be fed above its rated output voltage to maintain the output voltage.

The TLV75533 requires an output capacitance of 0.47-200 μF for stability. Use X5R- and X7R-type ceramic capacitors because these capacitors have minimal variation in capacitance value and equivalent series resistance (ESR) over temperature. Place a 1 μF or greater capacitor on the input pin of the LDO. Some input supplies have a high impedance. Placing a capacitor on the input supply reduces the input impedance [11].

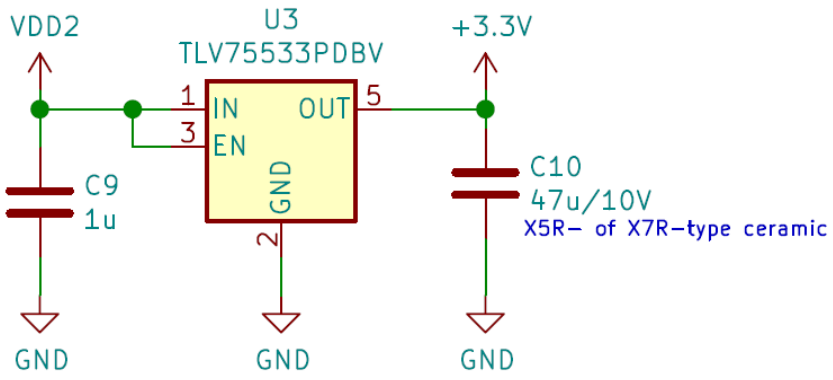


Figure 12: TLV75533 applied schematic

4.7 Load switches

To reduce power even further you can disable the sensors by shutting off their supply voltage if they're not needed. This can be done by load switches or relays. A load switch is a small electronic switch used to configure and manage power distribution. You can make a load switch with discrete components, but there are significant advantages to using a fully integrated IC load switch [12]:

1. **Saves space:** A key advantage of the integrated device over discrete load switches is the small footprint.

2. **Simpler design:** No discrete design time is required. Just buy the integrated chip and design it in to get all benefits and features.

Load switches also has several advantages over relays [12]:

1. **Quick output discharge:** An internal resistance in the form of a conducting MOSFET is connected across the load. It serves to discharge to the output capacitor quickly when the enable input goes low.

2. **Reverse-current blocking:** Some IC load switches implement a feature that blocks any reverse current. When the enable input goes low to turn off the switch, the reverse current block circuit is enabled

thereby reducing any potential reverse current from output back to input to a very low level.

3. **Saves power:** Using load switches to manage the power output of the supply minimizes power consumption, saves energy, and provides for longer battery life in portable applications. In its off state, the IC draws minimal quiescent current.

4. **Manages inrush current:** This feature prevents input voltage droop that may transpire when the switch is first turned on. This is the result of a high inrush current that occurs when charging the output capacitor. Most load switches provide a controlled ramp up of the output voltage as it charges the output capacitor.

5. **Implements sequencing:** Load switches let you implement the sequencing of loads during a turn-on or turn-off operation. Some designs with processors, FPGAs, and other chips require that different supply voltages be applied or disconnected in a specific order. Multiple load switches for each supply that are operated by the related embedded controller for timing meet this need. Certain load switches also feature an output signal that indicates when the output is fully turned on. This signal can be used in sequencing operations with other load switches in the system.

4.7.1 Selected load switches

The TPS22919 device is a small, single channel load switch with controlled slew rate. The device contains an N-channel MOSFET that can operate over an input voltage range of 1.6 V to 5.5 V and can support a maximum continuous current of 1.5 A. A few important specifications are [13]:

- Input operating voltage range V_{IN}: 1.6 V to 5.5 V
- Maximum output current $I_{OUT,MAX} = 1.5$ A
- Low power consumption:
  - ON-state $I_Q = 8$ μA (typical)
  - OFF-state $I_{SD} = 2$ nA (typical)
- On-Resistance R_{ON}: 89 mΩ (typical) @ $V_{IN} = 5$ V

The switch ON state is controlled by a digital input that is capable of interfacing directly with low-voltage control signals. When power is first applied, a Smart Pull Down is used to keep the ON pin from floating until system sequencing is complete. Once the pin is deliberately driven High ($> V_{IH}$), the Smart Pull Down will be disconnected to prevent unnecessary power

loss. The TPS22919 load switch is also self-protected, meaning that it will protect itself from short circuit events on the output of the device. It also has thermal shutdown to prevent any damage from overheating.

Figure 13 shows the application of TPS22919 for our 3 sensors. The resistor between VOUT and QOD pin is a quick discharge resistor to discharge the output capacitor when the enable pin goes low. For both in- and output capacitors, and the resistor, we used the recommended value as given in the datasheet [13].

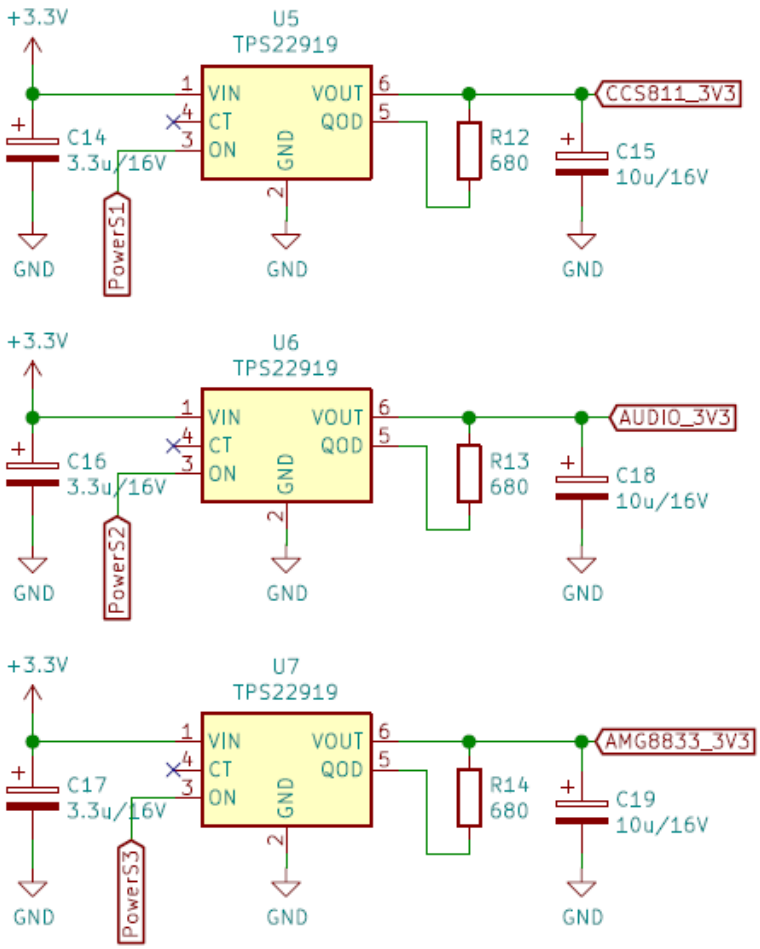


Figure 13: TPS22919 applied schematic

5 Microcontroller

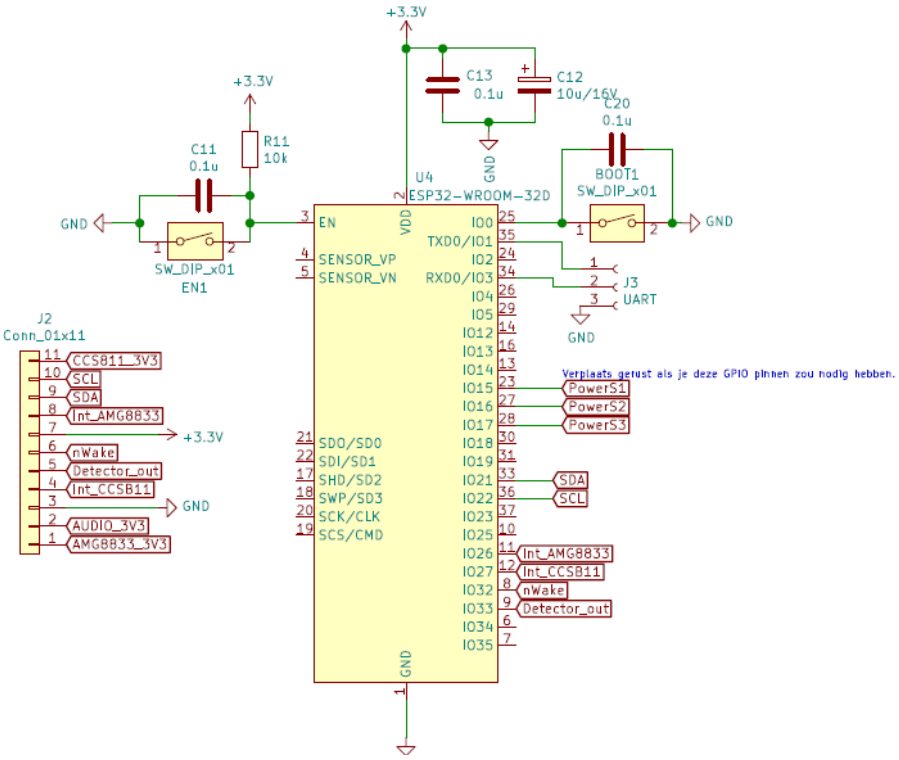


Figure 14: ESP32 applied schematic

6 Sensorboard

6.1 Design considerations

For the sensorboard design, we have decided to go for a separated board from the main board. This gave us the possibility to design it more freely. For the design, we have chosen to go for a two sided printed circuit board or PCB. All sensors will be placed on the top side of the board for it will be plugged onto the CPU board. Then, we looked what sensors are needed for the design. The parameters that need to be sensed are the sound level, air quality, temperature and the the presences of people. With this in mind we have chosen the following sensors:

- AMG8833 IR-sensor
- CCS811 air quality sensor
- BOB-12758 microphone breakout board

The main goal for the sensor board was to make it as low power as possible. As already mentioned above we have foreseen load switches to cut off the power to the sensors, when the system is non-active for a long period of time. Beside the load switches, we have studied the sensors to optimize the measuring sequence.

6.2 AMG8833 IR-sensor

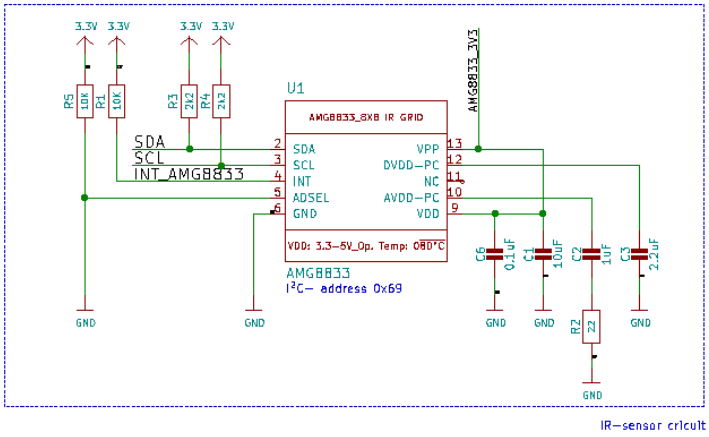


Figure 15: Schematic AMG8833

To detect human presence, we have decided to use an infrared sensor. The human body produces quite some body heat that is emitted. The AMG8833 is an 8x8-pixel IR-sensor, that maps the heat per pixel 16. This allows us to determine how many people are present in a certain area. For example we can determine if there is a single person present or a group of persons in a row.

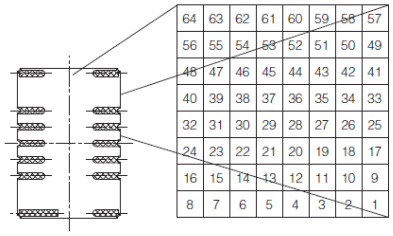


Figure 16: Heatmap as a pixel grid

One of the considerations that has to be studied is the effect of distance, ambient temperature and the field of view of the sensor. The sensor has

a total angle view of 60° for the horizontal and vertical axis of the pixel grid. Depending on the height, the area that the sensor covers, changes. If the height increases, the area also increases. Not only the field of view is influenced by the height but also the result of the measurements. The further the heat source is from the sensor, the lower the measured temperature will be.

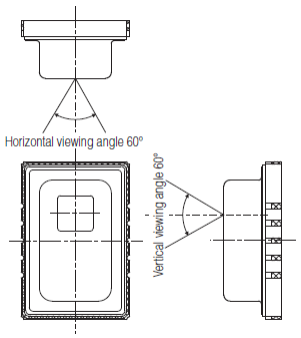


Figure 17: View angle AMG8833

The sensor operates in a temperature range of 0 °C up to 50 °C for the full humidity range between the 15 %RH and up to about 85 %RH. Heigher temperatures can be handles if the humidity stays below a certain value. This is illustrated below. Because, the system will be deployed in the cafeteria, this should not be a problem. An other aspect is the ambient temperature. The sensor has to be capable to determine the difference between the heat of a person and the heat of the surroundings. This will be tested and implemented in the software.

All communication to and from the sensor is done over I^2C. In addition to the I^2C-communication there is a AD-select input to set the I^2C-address, you can set the address to 0x58 or 0x59. There is also an interrupt output if you want to work with interrupts.

The last important aspect for this sensor is the power consumption. In the documentation it is stated that the sensor has a typical current of 4.5 mA in normal operation mode. When the sensor is brought to sleep the current drops to 0.2 mA. Our goal is to wake the sensor from sleep and when we have the measurement put it back to sleep as fast as possible.

6.3 CCS811 air quality sensor

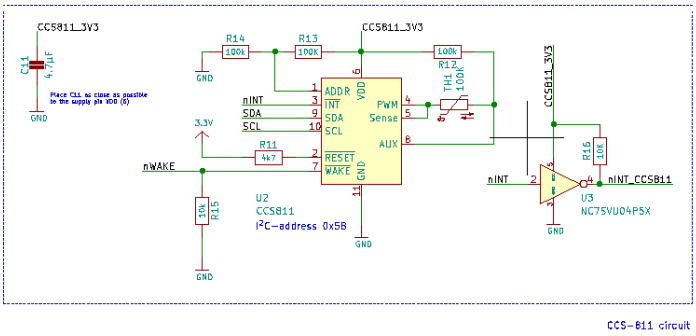


Figure 18: Schematic CCS811

For the air quality in the cafeteria, we have chosen to go for the CCS811 air quality sensor. The CCS811 is an ultra low power gas sensor with propable algortime system for detecting gasses. It has the ability to sense TVOC and an equivalent CO_2 concentration. This sensor has different measure modes which can help to make the application more flexible but also suitable for mobile equipment and battery powered systems. All settings and communication to the sensor is again done with I^2C. The I^2C-addresses are 0x5A and 0x5B, depending on the ADDR-pin. Besides the I^2C-communication there are some other signals that can control/monitor the sensor.

- There is a **ADDR**-pin for setting the I^2C-address to 0x5A or 0x5B
- The **nINT**-pin is an active low pin, that generates interrupts depending on the settings. This pin can generate interrupts when a measurement is available or when the equivalent CO_2 is above or under a certain value. This is a value that is best buffered before monitoring with a GPIO-pin.
- The **nReset**-pin is not only a reset pin but it can also be monitored to detect a software reset. This can also be an input. If this pin is pulled low you can reset the sensor.
- The **nWake**-pin is also an active low pin. This pin can be used to put the sensor into sleep just by pulling the pin low.

The following modes are possible:

- **Idle**: sensor does not perform a measurement: low current mode
- **Mode 1**: a measurement is performed every second: constant power mode

- **Mode 2:** a measurement is performed every 10 seconds: Pulse heating mode
- **Mode 3**: a measurement is performed 60 every seconds: Low power pulse heating mode
- **Mode 4**: a measurement is performed every 250 millisecond: constant power mode

The Idle-mode has a second purpose when switching from one mode to another. There is a requirement when changing the mode from one with a higher sample rate to one with a lower sample rate. If this occurs, the sensor has to be placed in idle mode for at least 10 minutes. However, this is not necessary in our application. When changing the mode from a lower to a higher sample rate this is not needed. In our application, we have chosen to search for one mode to operate in and always place the sensor back to sleep after the measurement has been read out.

In the documentation it is stated that the average current for a pulse cycle is 0.7 mA with a power supply of 1.8 V. If we calculate the power, we get the following result:

$$P = U \cdot I = 1.8V \cdot 0.7mA = 1.26mW$$

What is the power consumption of mode 3. And, if we compare it with the power consumption of mode 2 we can find:

$$\frac{P_{pulsemode}}{P_{lowpulsemode}} = \frac{7mW}{1.26mW} = 5.556$$

This is not expected, You can perform 6 times the measurements with a lower power consumption.

If we now perform the same comparison with mode 1 & 4, we get the following result:

$$\frac{P_{constantpower}}{P_{lowpulsepower}} = \frac{46mW}{1.2mW} = 38.333$$

The power consumption of the constant power modes is only a factor 38,333 bigger then it is in the low pulse power mode. While the amount of measurement is much higher in the constant power mode. This is something that could make our application more low power. This will be investigated by studying the measure behaviour of the sensor. To make it possible that we could use he interrupt functionality of the CSS811, we have foreseen a buffer with an operational amplifier for the nINT-signal. This is foreseen in the prototype and if this is eventually not implemented in the final design, this

can be removed. The nINT is not an essential connection for the CSS811 to work properly.

6.4 BOB-12758 microphone breakout board

We have informed us by colleague students who had used a microphone module in the first semester. One group came with a breakout board with a microphone and a triple operational amplifier circuit. The proposed breakout board could do several things, one of them was peak detection. For our application, there is only the need to check the average sound level, so we only needed a microphone and one operational amplifier. In figure 19, we can see the schematic of the breakout board.

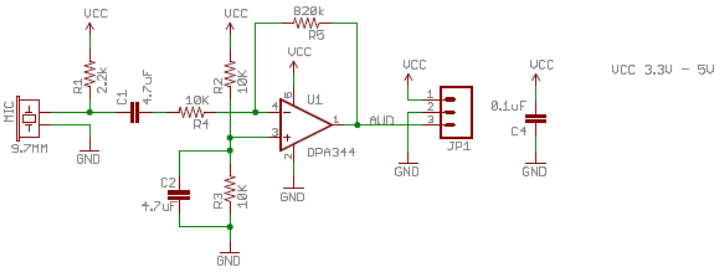


Figure 19: Schematic BOB-12758 microphone breakout board

Because we wanted to have the average sound level, we decided to place a detector behind the opamp. The second reason to use the detector is that the ESP32 can sample up to about 27 ksps. By just looking at the sample rate, it is clear that the ADC is not fast enough to fulfil the Nyquist theorem.

The detector needs time to come from the low level to the sound level. Finding the best value of τ will be one of the investigations. This will be determine along with the test of the other sensors. The microphone must receive power for about 10 times τ. We know from the theory that after 5 τ, 97% of the applied voltage is reached. The extra 5 τ is chosen for two reasons. Firstly to compensate the rectified signal by the detector and secondly to have a stable value of the average sound level.

6.5 Board design

When designing the board, we have started from the basic schematics of the provided break out boards of the sensor boards 15 18. With some extra guidelines from the documentations of the sensors. For the microphone sensorboard as mentioned, we needed to design a detector circuit. We have chosen to go for a simple detector with a diode followed by an RC circuit, see Figure B.

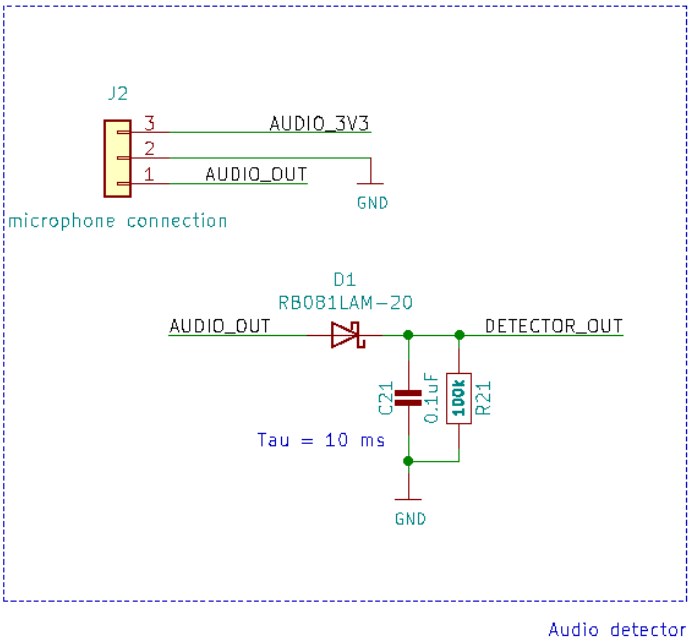


Figure 20: Schematic simple detector

During the documentation of the circuit it was noticed that the audiosignal of the breakout board is superpositioned onto a DC-voltage that is the half of the supply voltage. This can be seen on figure 19. The resistor circuit R2 and R3 will divide the supply in half, so half of the supply voltage will be available at the non-inverting input of the operational amplifier.

In the prototype of the board, this was not taken into consideration. This can be solved in different ways not only by hardware but also by software. If we want to solve this with hardware without loosing information, we can just put several diodes in series until we come to a sum of the forward voltages that is the half of our voltage supply. The power supply is $3.3V$. This method will not increase the accuracy of the measurement so it is not an improvement.

$$V_{DCoffset} = \frac{V_{CC}}{2} = \frac{3.3V}{2} = 1.65V$$

This $1.65V$ can approximately be obtained by placing 2 Schottky diodes with $V_f = 0.45V$ and a normal diode with $V_f = 0.7V$ in series.
$V_{ftotal} = 2 \cdot V_{fschottkey} + V_{fnormaldiode} = 1.6V$
This will almost eliminate all the DC-voltage in the audiosignal. But still, there could be some DC- voltage left or some information on the amplitude could get lost.
A better approach to the problem would be to solve this with software.

Then, there is no chance that any information of the sound gets lost expect the information below $00.45V$, but also this could be solved by providing an own DC-offset on top of the audio signal. To solve it with software, we need to know what the ADC value would be if no sound was detected. The DC-voltage after the detector will be $0.45V$ lower because of the Schottky diode.
$V_{DCoffset} = \frac{V_{CC}}{2} - V_{fschottkey} = \frac{3.3V}{2} - 0.45V = 1.2V$
The reference level for no sound is $1.2V$. The maximum audio voltage level would be $2.85V$. What sound level this would be is unknow to us. Because, it is more a relative perception with a visual indication, exact values are not necessary. The DC-offset has a positive effect on the detection of the sound level. Because the value is above the forward voltage of the diode we do not loose information on the amplitude. But as mentioned we need to do a comparison with the reference level. We can set some thresholds for indicating the experience of a certain value. For example, if the sound level is above a certain value, it is not pleasant to study in the cafeteria.

To determin the optimal value for the resistor and the capacitor of the detector, we would wait to look at the other response time of the other two sensors. We have taken a initial value of 1 ms. This can be realized with a resistor of $10k\Omega$ and a capacitor of $0.1F$.

For the other two sensors we have as mentioned based our schematic on the schematic of the break out boards. We only needed one extra component and that was an operational amplifier to buffer the nINt of the CCS811. This was recommended by the documentation of the CCS811. Also, if this interrupt function is not desired, we can just remove this part of the circuit and the system will work just fine. We have brought all the other connections of the sensors to a central header of 11 pins, so the board can be plugged onto the CPU board. The break out board of the microphone will be placed on top of this board by a 3 pins header. The schematic drawing of the sensor board can be find the appendix **??**.

Unfortunately during the prototype phase, the pitch of the pinheader was not found, we have based the pitch according to a recommended connector of the dealer. Afterwards, we came to the idea to check the value at the developer of the breakout board and found a recommended pinheader with a different pitch. The drawing has been updated. Also, this has not been verified because of the corona situation.

For making the system as low power as possible we have an exclusive power supply for every sensor, so we can control the supply for every single sensor separately. So, the system is flexible depending on the need of what stimulus needs to be monitored.

6.6 Investigations

In this section, we are reviewing all our investigations on the sensors. The last part of this section will be the conclusion of all results and how we will implement the sensors in the system.

Time needed for measurement is available

We started to do some measurements on the **CCS811** what is the difference in the following to situation:

- Time needed for initialization
- Time needed for the sensor to wake from sleep

The first situation is to see if we could save some energy if we shut down the power supply of the sensor. Also, this is interesting to know if we use the load switches to turn of the power of each sensor. The second situation is as the CCS811 is normally used in an application. The test procedure for these tests was as followed. After the sensor was waked or initialized, a timer was started. After every ms, we looked in the registers of the sensor if a measurement was available. This was done with the following method **Adafruit_CCS811::available()** from the**Adafruit_CCS811**. If we get **True** back from this method, we stop the timer and made it visible.

We got the following results from these test.

- **Time after initialization:** It takes more than several sample period before we get a result. Also in the documentation, it is stated that after initialization the results are not that reliable.
- **Time after sleep:** About the sample rate of the mode that the sensor is set in. the shortest time is 250 ms in **mode 4**.
- **Time started from idle:** It took more time before the first measurement was available when changed from idle to the wanted mode.

Conclusion:
The first conclusion of these tests is that bringing the sensors out of sleep is the fastest way to get a measurement. This is because for getting the sensor out of sleep we just need to alter the mode.
Second conclusion, the sample period has always been respected. It is not the case that the sensor will perform a measurement at wake up. If the sensor is woken up, it will take the time of the sample period for a measurement to be available.

The second sensor, we needed to check was the **AMG8833**. The procedure was the same, the only difference is, to bring the AMG8833 out of

sleep, this needs to be done with an I^2C-command instead of a GPIO pin. Also, for this we use the library **SparkFun_GridEYE_Arduino_Library**. The method to wake the sensor is wake() and putting it back to sleep with sleep(). We have chosen to use the sample rate of 10 samples per second. For reading out the raw data of a single pixel there is the method "getPixel-TemperatureRaw". We checked every millisecond if the first pixel data was available. We noticed during testing that not all pixel data was available after 100ms. So we changed the check so that the timer may be stopped if the first pixel and the last pixel of the array had a value. With this measurement principle, we found that after 105 ms the data of all pixels was available.

Before going further with the last sensor, we wanted to see if we put the the readings after each other what the effect would be. How much time it cost to read out the AMG8833 sensor. We made a method that read both sensors after each other. We found that the CCS811 was ready to read out. What this means for the implementation will be explained further on in this report.

Now, we have come to the last sensor the microphone, we have chosen to use a breakout pcb as mentioned above. The only thing, we needed to check is which τ we can take for the filter of the detector. We could not test this because there was no hardware available for this. Based on the previous results, there is a delay of 105 ms between waking the sensors and have the measurements available. As mentioned, we wanted a period of 10 τ beacause the detector has to reach a stable average value to measure. If we divide 105 ms by 10, we get a τ of 10.5 ms. If we round this to 10 ms, we can realized this τ with a resistor of 100 kΩ and a capacitor of 100 nF.
$\tau = R \cdot C = 100k\Omega \cdot 100nF = 10ms$.
Which are values that are easily found as SMT-components. If we look more deeply into the 10.5 ms for τ, this is also interesting for audio in another way. All frequency higher then 100 Hz will be detected quite enough. Only some lower frequencies will probably not be detected. But if we look at the chart of equal loudness for the different frequency at different sound pressures. We can see that for frequencies lower then say about 400 Hz the amount of sound pressure needed for a same loudness experience is much larger.

With this substantiated scientific information, we can state the value of 10 ms is a good value for a quick and correct response for indicating the loudness of noise in the cafeteria. If we also take a look at the frequency response of the breakout board. The signals from 100 Hz will be picked up to about 1 kHz with the same gain value. Outside these boundaries, the gain of the audio signal will fall back. The value of 10 ms is again a good value for the τ.

Human presence detection

For this we have done two straight forward experiments with the **AMG8833**.

- **First experiment:** Heat detection on a person
- **Second experiment:** Detecting a person from a height

For the first experiment we have put a person in front of the AMG8833 at about 20 cm, 40 cm. Then measurements were performed when pointing the sensor at the face, a lined body part and the crown of the head. There was some difference in the measured values. The difference between the lined body part and the face was noticeable. There is a small difference between the face and the crown but not that much.

This concludes, we can detect a human by the radiation coming from the crown of the head.

With this confirmed, we progressed into the second experiment. Is it possible to detect a human below the sensor at a height of 3,4 metres (possible to test at home during corona pandemic). The height difference between the head of an average standing person and the sensor is about 1,7 metres. At this height the sensor will covers an area of $3.85m^2$. We know the sensor had a angle view of 60°. If we then calculate according to the principal geometric drawing 21, we find:

$x = height \cdot tan(30°) = 1.7m \cdot 0.577 = 0.981m$

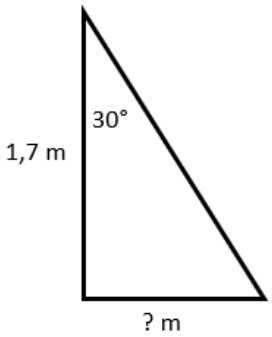


Figure 21: Prinicipal drawing view sensor AMG8833

Finding the field of view of the sensor at this height, **x** must be doubled and squared:
$A = (2 \cdot x)^2 = (2 \cdot 0.981m)^2 = 3.85m^2$

If the sensor is at a height of 3.4 metres the covered area will be 3.85 m^2, which is a workable area for noticing someone is sitting at a table. The area will be even bigger because the distance will be bigger. We wanted to know if it is possible to detect a person and to track the person. For this we have placed the sensor at this height and analysed the results if someone moved beneath the sensor. When analysing the result we found that a person was represented by 1 or 2 pixels reffig:human detectionwith a value that is about 2-3 °C higher than the surroundings.

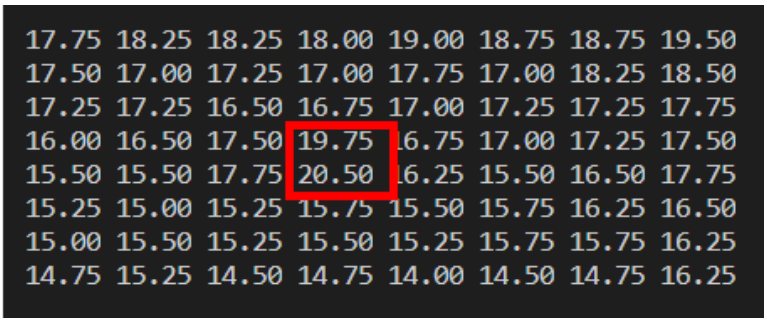


Figure 22: Human detection at a height of 3.4 metre

The test was performed in a space with an overall temperature of about 17 degrees on ground level. Which is not a normal room temperature. During testing the results were fluctuating about 0.25 °C which is the resolution of the sensor in °C. Notice the higher numbers at the top pixels. This coud be because the test setup was attached on a wooden staircase at 3.4 metres high where some licht bulbs hang.

Conclusion: It is possible to detect a person at a height of 3.4 m in a room with a temperature of 17 °C. It was also possible the follow his movements. If placing this into practice The best way is to compare the value of a pixel top an overall temperature or to a reference value. This should have been tested at the cafeteria during optimization of the system.

6.7 Programming the sensors

With the experiments mentioned above we made some methods for initializing the sensors and getting the measurment results of the sensors. For having a clear overview of our code we have made a own library with these methods, that combines our own code with code from the already mentioned libraries.

We can divide the created methods in two types: initializations methods, for initializing the sensors after power on and methods for reading the sensor data. We have foreseen for the initialisation a main method were the different methods will be called one after each other with the correct flow.

The slowest sensor will be initialized first so the time to start up can be use to initialize the other sensors.

For the initialization of the sensors, we just enable the I^2C-communication, set some GPIO-pins, set the ADC and set some sensor registers with the methods of the libraries. For setting, the GPIO's and the ADC we use the standard libraries of the system **driver/gpio.h** and **driver/adc.h**.

For the methods for retrieving the data from the sensor. The method is made in the following steps:

- Wake the sensors
- Wait until data available
- Retrieve data
- Put the sensor back to sleep

7 Wireless connectivity

7.1 Wi-Fi connection

In order to transmit the sensor data to the server, the Wi-Fi connectivity of the ESP32 is used. The transmission is provided by a TCP connection that is established between the ESP32 and the python server. At the initialisation of the python server, a socket is opened, this socket is used to receive the data. In this manner the server is always listening on the socket, until a message is received. The ESP32 sends the data in the following order: pixeldata (64 floats), temperature (1 float), audio level (1 float), CO_2-level (1 unint16), TVOC-level (1 unint64) and an ID (1 uint8). The received data is processed by order to determine which data is represented by the received values. The data is parsed into different variable, these variables are then used to update the values in the database.

7.2 Power consumption

To determine the power that is consumed when the sensor data is transmitted once, a measurement is performed. The result is displayed in figure 23.

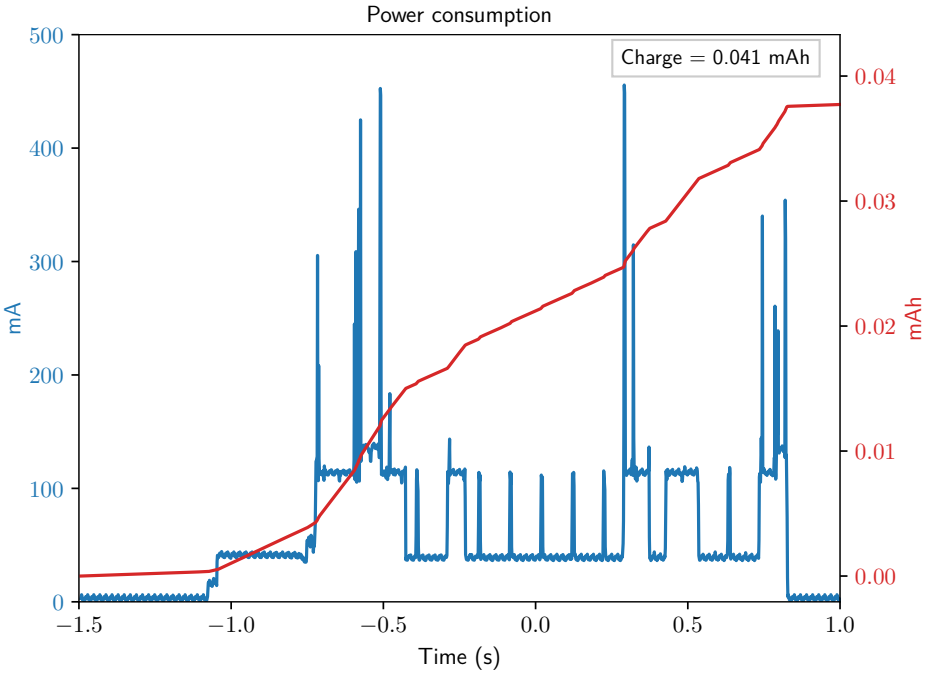


Figure 23: Power consumption of one Wi-Fi transmission

In order to interpret the current consumption, the used charge is displayed in [mAh]. This results in 0.041 mAh of charge that is used in order to transmit the sensor data once. With this knowledge, we can calculate the amount of Wi-Fi transmissions that are possible with the 6400 mAh battery:

$$\frac{6400}{0.041} \approx 156097 \text{ transmissions}$$

If there are 6 transmissions per hour and the system is active for 7 hours a day, this leads to a battery life off 3716 days. However, this calculation only accounts for the Wi-Fi transmission, other power consumption such as stand-by power, CPU power and sensor power are not accounted for in this calculation.

8 Software design

Functionality

- Pull sensors from sleep state every 10 minutes and read sensor values
- Send sensor data to server, over Wi-Fi

- Perform periodic synchronisations to get time information
- Use time information to disable power to sensors and put CPU in deepsleep during the night

Overview

The software is based on a number of states these states are: NTPsync, checktime, nightsleep, wakefromnightsleep, wakesensors, readsensors, senddata and sleep. An overview of the complete software and the transitions between the states is illustrated in figure 24. The states are discussed further below.

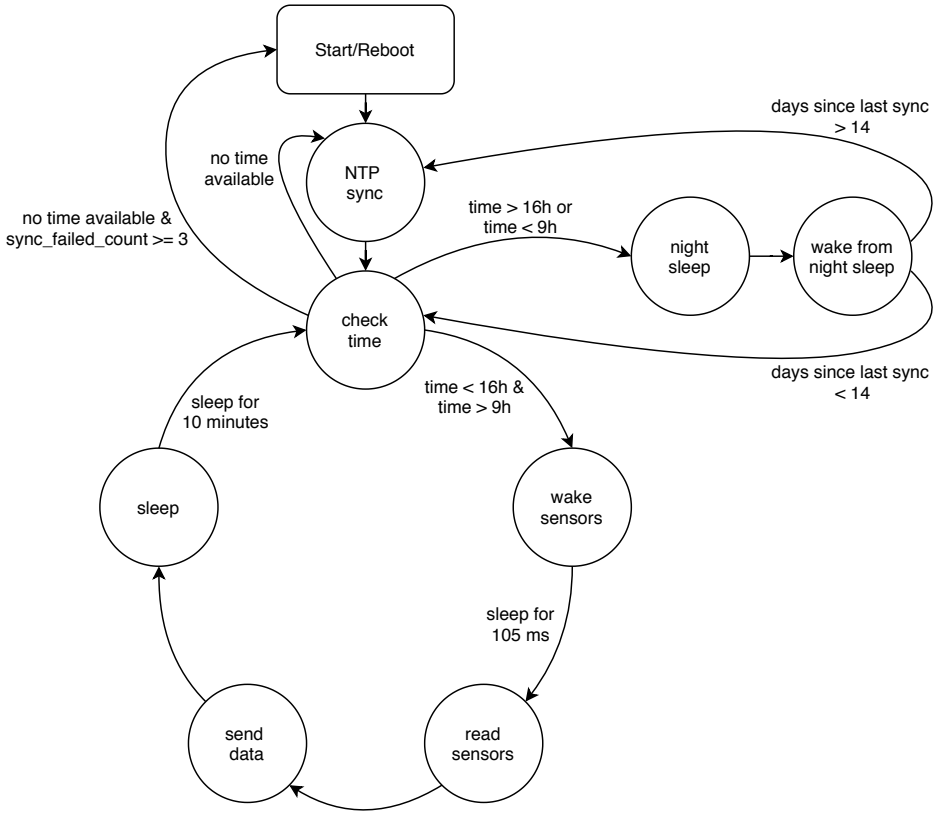


Figure 24: State diagram

NTP-sync

In this state an NTP synchronisation is performed. This provides time information which is stored in the RTC memory and maintained by the RTC clock. The time information allows the system to know when it is night time. During the night the sensors can be disabled, using the load

switches, and the microcontroller can go in to deepsleep until the morning. This way the sensors only receive power between 9 AM and 4 PM.

Check time

This state has two functionalities. Firstly the current time of day is retrieved from the RTC clock. If this time info is not available, the NTP synchronisation was not successful. When this is the case the microcontroller goes in to deepsleep for five minutes and the next state will be the NTP sync to retry the synchronisation. If the synchronisation fails three times, the system is rebooted. Next to this, when the synchronisation was successful, the time information will be available. This time information is then used to check the current time. If the time is between 9 AM and 4 PM the system remains active, thus the next state is set to wakesensors. If the time is after 4 PM and before 9 AM, the system has to go to sleep, thus the next state is set to nightsleep.

Night sleep

This state is entered when it is past 4 PM and before 9 AM. During this time the sensors are disabled and the microcontroller goes in deepsleep. The sensors are disabled by raising the GPIO pins which control the load switches. When the GPIO pins are high, the load switches cut off the power to the sensors. Before entering deepsleep, the next state is set to wakefromnightsleep. During the night, no sensor data has to be collected and transmitted because the campus is closed. So by cutting off the power to the sensors and putting the microcontroller in deepsleep during the night the power consumption is drastically reduced.

Wake from night sleep

When the system wakes up at 9 AM, the power to the sensors has to be enabled. This is done by lowering the GPIO pins which control the load switches. Next to this, the system checks when the last synchronisation was performed. Because the RTC clock is derived from an internal RC oscillator with a frequency of 150 kHz and an accuracy of +/-5% the clock time will drift away from the actual time. In order to prevent this, an NTP synchronisation has to be performed periodically. In order to retain an accurate clock, the synchronisation is performed every 14 days. Thus if the last synchronisation was performed 14 days ago, the next state is set to NTPsync. If the last synchronisation was more recent, the next state is set to checktime.

Wake sensors

In the wake sensors state, the sensors are taken out of their sleep state and put in active mode. When the sensors wake from their sleep state the sensor values can not be read immediately. A waiting period of 105 milliseconds is necessary in order to have stable sensor values. During these 105 milliseconds the sensors remain active but the microcontroller is put into deepsleep mode. Before entering deepsleep, the next state is set to readsensors.

Read sensors

In this state the sensor values are read out. Once the data is acquired, the sensors are put back in to their sleep state and the next state is set to senddata.

Send data

In this state, the sensor data is transmitted to the python server which is running on a raspberry pi. Once the transmission is successful or the connection has timed-out the next state is set to sleep.

Sleep

Once the sensor data has been transmitted to the server, the sleep state is entered. In this state, the microcontroller is put into deepsleep for 10 minutes. Before entering deepsleep, the next state is set to check time.

9 Autonomy

9.1 Consumption sources

In this section the different sources that consume power will be discussed.

Wi-Fi communication

The system transmits the sensor data to the server over a Wi-Fi connection. This data transfer happens every 10 minutes. Because the sensors are disabled between 4 PM and 9 AM, no Wi-Fi connection is needed during that time. With a wake time of seven hours and 6 transmissions per hour, this leads to 42 transmissions per day. A measurement of the typical power consumption for one Wi-Fi transmission is illustrates in figure 25. The charge that is used for one transmission is 0.041 mAh. This leads to the following consumption per day:

$$42 \cdot 0.041 = 1.722 \text{ mAh/day}$$

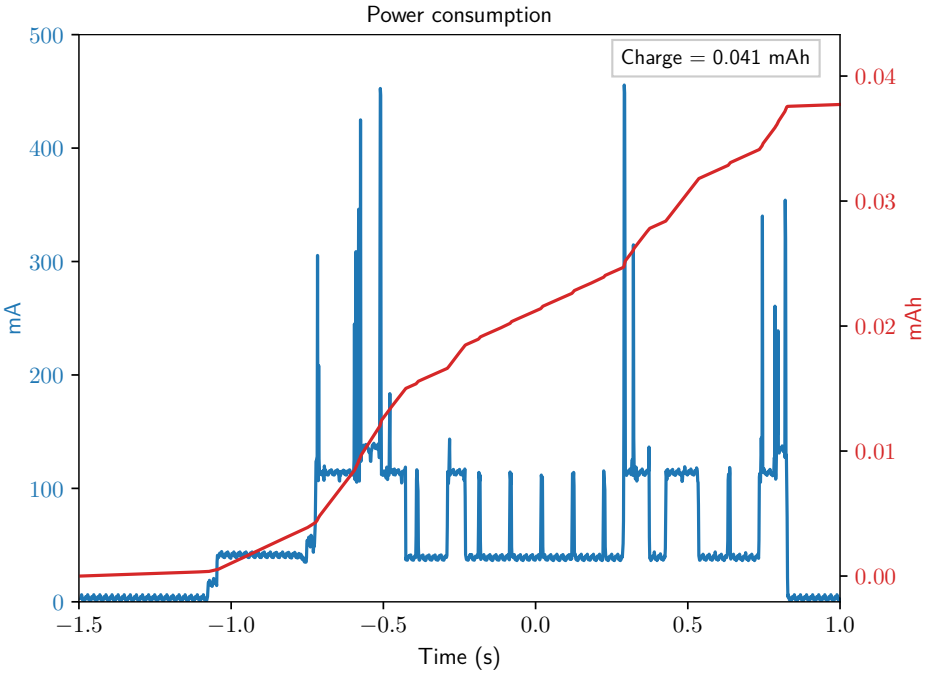


Figure 25: Power consumption of one Wi-Fi transmission

Sensors

The sensors are read out every ten minutes. With a wake time of 7 hours[5], this leads to 42 sensor measurements per day. The power consumption to read out the sensors once is illustrates in figure 26.

[5]Only awake between 9 AM and 4 PM

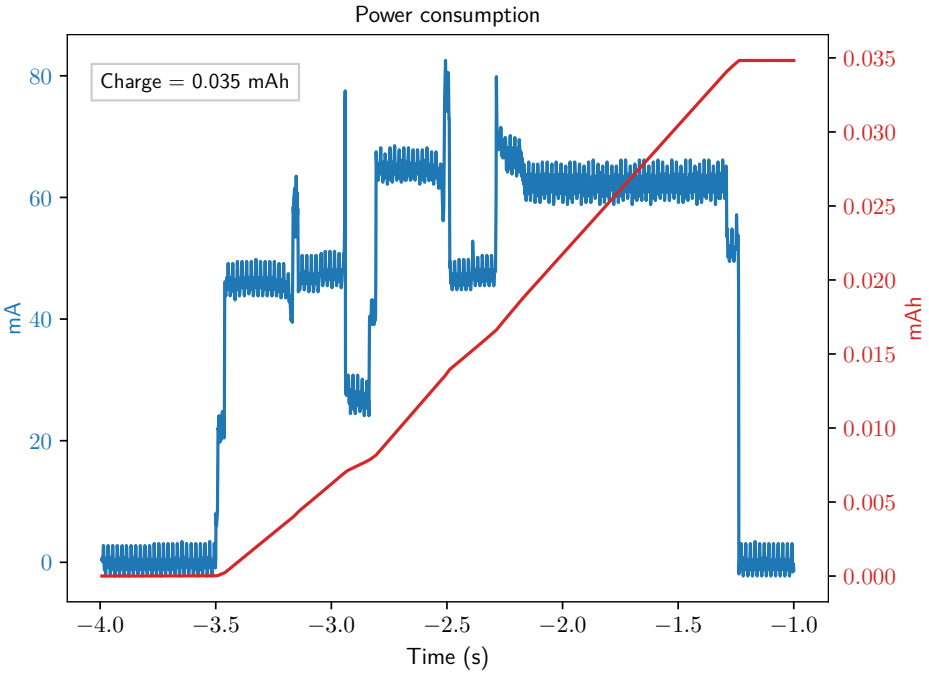


Figure 26: Power consumption of one sensor measurement

The charge that is used to read out the sensors once is 0.035 mAh. This leads to the following consumption per day:

$$42 \cdot 0.035 = 1.47 \text{ mAh/day}$$

NTP synchronisation

In order to synchronise the RTC clock an NTP synchronisation has to be performed periodically. This synchronisation is performed once every 14 days. The power consumption for one synchronisation is illustrates in figure 27.

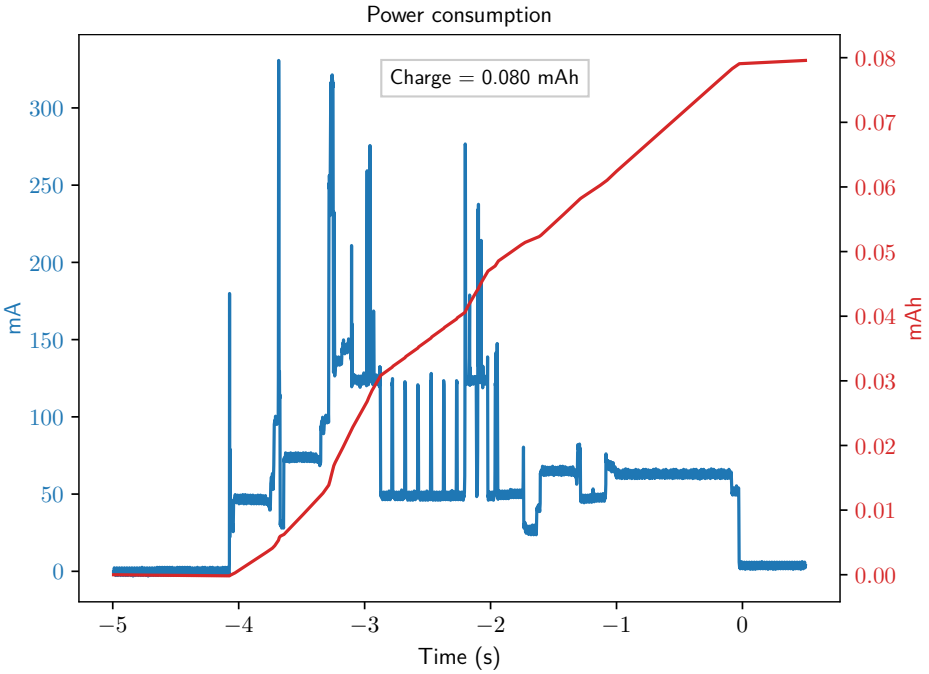


Figure 27: Power consumption of one NTP synchronisation

The charge that is used for one synchronisation is 0.08 mAh. However, this synchronisation only happens once very 14 days, this leads to the following consumption per day:

$$\frac{1}{14} \cdot 0.08 = 0.0057 \text{ mAh/day}$$

Standby daytime

[6] Between 9 AM and 4 PM the system is 'active', this means that the sensors receive power. However when the sensors are not being read out they will be put in their respective sleep modes. The theoretical consumption that is gathered from the datasheets is listed below:

- AMG8833 sleep mode: 0.2 mA
- CCS811 sleep mode: with $V_{dd} = 1.8\,\text{V}$: 19 μA
  $\Rightarrow$ with $V_{dd} = 3.3\,\text{V}$: $\frac{19}{1.8} \cdot 3.3 = 34.83$ μA

[6] No measurement were performed to determine the stand-by power, this has two reasons. 1: the development board to test the software has an onboard LED which increases the power consumption so the gathered measurements wouldn't be representative. 2: to measure the power consumption a $1\,\Omega$ resistor was inserted between the power source and the development board. This resistor is used as a sense resistor but can not provide accurate µA measurements due to the noice that is introduced.

- microphone: the power to the microphone is only enabled just before the measurement because of this the microphone has no stand-by/sleep consumption.

Next to the consumption of the sensors, there is the sleep consumption of the microcontroller. If the microcontroller is not active, it is put in deepsleep. In deepsleep there is a consumption of 10 μA. Next to this consumption, two GPIO pins have to remain high during deepsleep: one to keep the CCS811 in sleep (nWake) and on to drive the load switch to disable the power to the microphone. The GPIO pins use an internal pull-up resistor of 45 kΩ this leads to the following consumption for one GPIO pin:

$$\frac{3.3\text{V}}{45000\Omega} = 73.33\ \mu\text{A}$$

For both GPIO pins this results in a consumption of 146.67 μA.
For 7 hours a day the system is in active mode, this results in a charge consumption of:

$$7 \cdot (0.2 + 0.03483 + 0.01 + 0.1467) = 2.74\ \text{mAh/day}$$

Standby nighttime

[7] Between 4 PM and 9 AM the system goes in to sleep mode. In sleep mode no sensor measurements are performed and no data is transferred via Wi-Fi. To reduce the power, the power to the sensors is cut-off by using the load switches. The remaining stand-by/sleep consumption originates from the CPU consumption in deepsleep and the three GPIO pins that are kept high to drive the load switches. This leads to the following consumption:

$$10\ \mu\text{A} + 3 \cdot 73.33\ \mu\text{A} = 230\ \mu\text{A}$$

The system remains in sleep mode for 17 hours a day this leads to the following consumption per day:

$$17 \cdot 230\ \mu\text{A} = 3.91\ \text{mAh}$$

9.2 Total power consumption

In the previous section all the different sources that consume power were discussed an overview is given in table 2.

[7] No measurement were performed to determine the stand-by power, this has two reasons. 1: the development board to test the software has an onboard LED which increases the power consumption so the gathered measurements wouldn't be representative. 2: to measure the power consumption a 1 Ω resistor was inserted between the power source and the development board. This resistor is used as a sense resistor but can not provide accurate µA measurements due to the noice that is introduced.

Source	**Charge per day [mAh/day]**
Wi-Fi	1.722 mAh/day
Sensors	1.47 mAh/day
NTP sync	0.0057 mAh/day
Standby daytime	2.74 mAh/day
Standby nighttime	3.91 mAh/day
Total	9.85 mAh/day

Table 2: Overview of the power consumption per day

There are two Li-Ion cells that are used to power the system, they both have a capacity of 3200 mAh which results in a total capacity of 6400 mAh. This leads to the following battery life:

$$\frac{6400}{9.85} = 649 \text{ days}$$

10 DashBoard

Functionality

- Recieve the data from the sensor
- Store the data in a database
- Display an overview with the avarages of every sensor
- Display the data from each sensor on a heatmap

Overview

The intention of this dashboard is to display the data that the different sensors on all the devices in the cafeteria generate. It gives an overview of what's happening in the cafeteria in an airquality, temperature and sound perspective. The infrared camera data which is an 8 by 8 pixel picture is drawn on a heatmap. This gives a nice overview of the busy and the less busy area's in the cafeteria.

10.1 Backend

The backend is what runs on our server, in this project a raspberry pi was chosen as it's an easy to use and small computer that can easily be set up to be used as a webserver, apache2 was used in this project to host our dashboard. Myqsl is used for the database that stores the sensor data.

Recieve data

To recieve the data a python script (WiFi.py) is used that listens to the 8091 port, this is the port that the device sends his payload to. The protocol that is used to send data from the device to the raspberry pi is TCP. A more in depth explanation can be found in Wi-Fi connection.

Store data

The recieved data is stored in a mysql database, this is also implemented in the WiFi.py script that is used to recieve the data. To use the database in the python script a mysql.connector is used. The database uses 3 tables:

- location: this table is used to store the location of the sensor. It has an x-, y-coordinate and the location id.
- sensor: this table stores the sensor id and has the location id as a foreign key.
- readings: This is the table that stores all the sensor data with the date and time that it was recieved. All the data that isn't an array is stored as a float, the arrays with the 64 pixels from the infrared camera are stored as a json string. The two other columms in this table are the reading id and the sensor id.

A visual representation of the database descript above can be seen in figure 28.

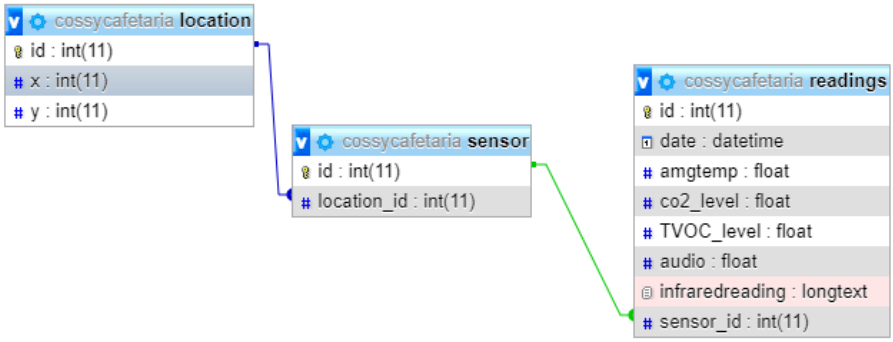


Figure 28: Visual represntation of the database

Data from database to dashboard

A php script is used to serve the data to the frontend of the dashboard. This php script is called database.php. It's a simple script that connects to

the database on the raspberry pi and gets the last recieved reading from all the different devices. Then all this data is encoded as json and passed on to the website, this website is the dashboard.

10.2 Frontend

The frontend is the part of the dashboard that people can interact with and view the data from all the different devices and sensors. This website has 6 different pages, an overview page that shows the averages of all the sensor data from all the different sensornodes. The other pages show a heatmap with the specific sensor data of each sensornode seperatly drawn on it. For example, there is an activity page that display where the infraredcamera picks up heat signals, a temperature page that displays where it is warmer or colder. Each type of sensor reading has it's own page.

Overview page

As already said in the short description above, this page takes the average of all the sensor data from the different sensornodes. On this page two subsection are made, the first one is named activity and under it the busyness of the cafetara is show by giving the average of all the pixeldata from all the sensornodes. In the second subsection the sensor data related the the environment is shown, the numbers desplayed here are also an average taken from all the diffrent sensornodes. The displayed data here is: Co2 level, TVOC level, Sound and temperature. Al this can be seen in figure 29.

Figure 29: Screenshot from the overview page

The avarages of all the data are calculated using a javascript script in

the website, so each time there is new data and the webpage is reloaded this wil update to the latest data. The javascript loops through the data from the database that it get's in the JSON format from the database.php backend script.

Page with heatmap

These pages show one single type of data from that the sensor node gathers, in this case the pixeldata from the infraredcamera. On figure 30 you can clearly see the different elements of this dashbaord page. up top there is the Cosy Cafetaria title with the menu to navigate all the different pages below it. Under this menu is where the actual data is displayed on the floor plan of the cafeteria. This floor plan is loaded in as an image that is used as background to draw the heatmap over. For the heatmap a javascript library is used called heatmap.js, this library makes it easy to add the datapoints with there value and location and draw them ontop of the floor plan. It also draws a legend in the botom right corner.

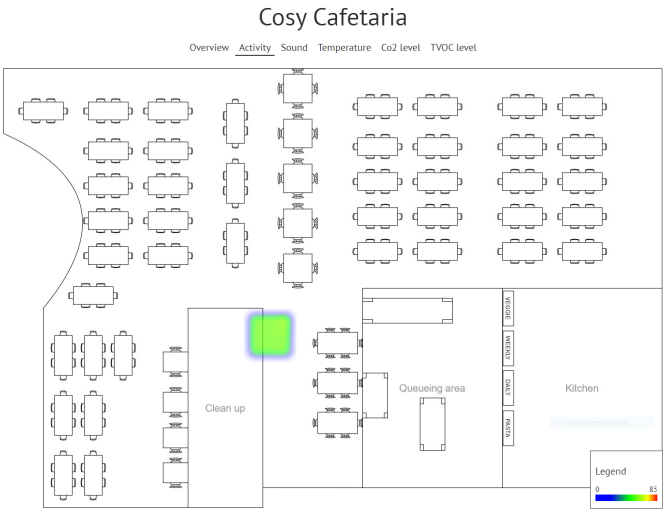


Figure 30: Screenshot from the activity page

If the cafeteria would be fully equiped with our sensornodes a completer map could be drawn on which you could clearly see the heatsignature of every person in the cafeteria. This would make it easy to see if your favourite spot was still available or not and how busy it is in the queue.
On the other pages of the dashboard the other sensor data is displayed in the same way. The temperature data would make it easy to spot where it's is hotter or rather where it's colder. Same goes for the other data that is collected by the sensornodes.

11 Case Design

For our system, we have designed a simple case for building in the system. We have made the design with fusion 360. This program has the abiliy to import different types of 3D-files. By importing all our seperate parts of the design and align them we could make a case that fits with some holes so the sensors have a opening above them to measure the different stimulus. The design of the case is not a very complex design but there have been taken some important design considerations. For the opening above the AMG8833, we have made the opening a little bit wider so it will not cover parts of the field of view of the sensor. Also, to hang the design to the ceiling we have drawn four loops at the upper side of case so we can hang the case with a cable to the ceiling everywhere the systems needs to be placed.

Possible improvements

The case could be made more closely to the the internal parts. Specific at the AMG8833 and the CCS811. For the CCS811 this is not really an issue because it is a gas sensor but for the AMG8833, the case could block the view of the sensor if the opening is not big enough. The weight distribution is not optimal, the batteries are at one side of the design, so the balance is not even spread through the design. This could result in a not perfect horizontal hanging case. We could adapt the string to balance out the weight so it will hang horizontal.

12 Financial estimate

Kostprijs van het project opstellen.

13 Validation and measurements

Hier moeten we aantonen of dat alles werkt. Of eventuele fouten verklaren.

References

[1] "Wat je moet weten over lithium ion accu's," https://www.ledscherp.nl/blog/blog/wat-je-moet-weten-over-lithium-ion-accus.

[2] Wikipedia, "Lithium-ion battery," https://en.wikipedia.org/wiki/Lithium-ion_battery.

[3] *NTR2101P, Single P-Channel, Small Signal, SOT-23*, ON Semiconductor, June 2019, datasheet.

[4] *BQ2407x Standalone 1-Cell 1.5-A Linear Battery Charger with PowerPath*, Texas Instruments, September 2008, datasheet.

[5] S. M. Dúlcey, "How much power does a usb port put out?" https://www.quora.com/How-much-power-does-a-USB-port-put-out.

[6] *BQ297xx Cost-Effective Voltage and Current Protection Integrated Circuit for Single-Cell Li-Ion and Li-Polymer Batteries*, Texas Instruments, March 2014, datasheet.

[7] *FDS9926A, Dual N-channel 2.5 V specified PowerTrench MOSFET*, Fairchild, July 2003, datasheet.

[8] *ESP32-SOLO-1*, Espressif, 2019, datasheet.

[9] B. SCHWEBER, "Both switchers and ldos have their place in power distribution, as each has special attributes." https://www.powerelectronictips.com/use-ldo-versus-switching-regulator-faq/.

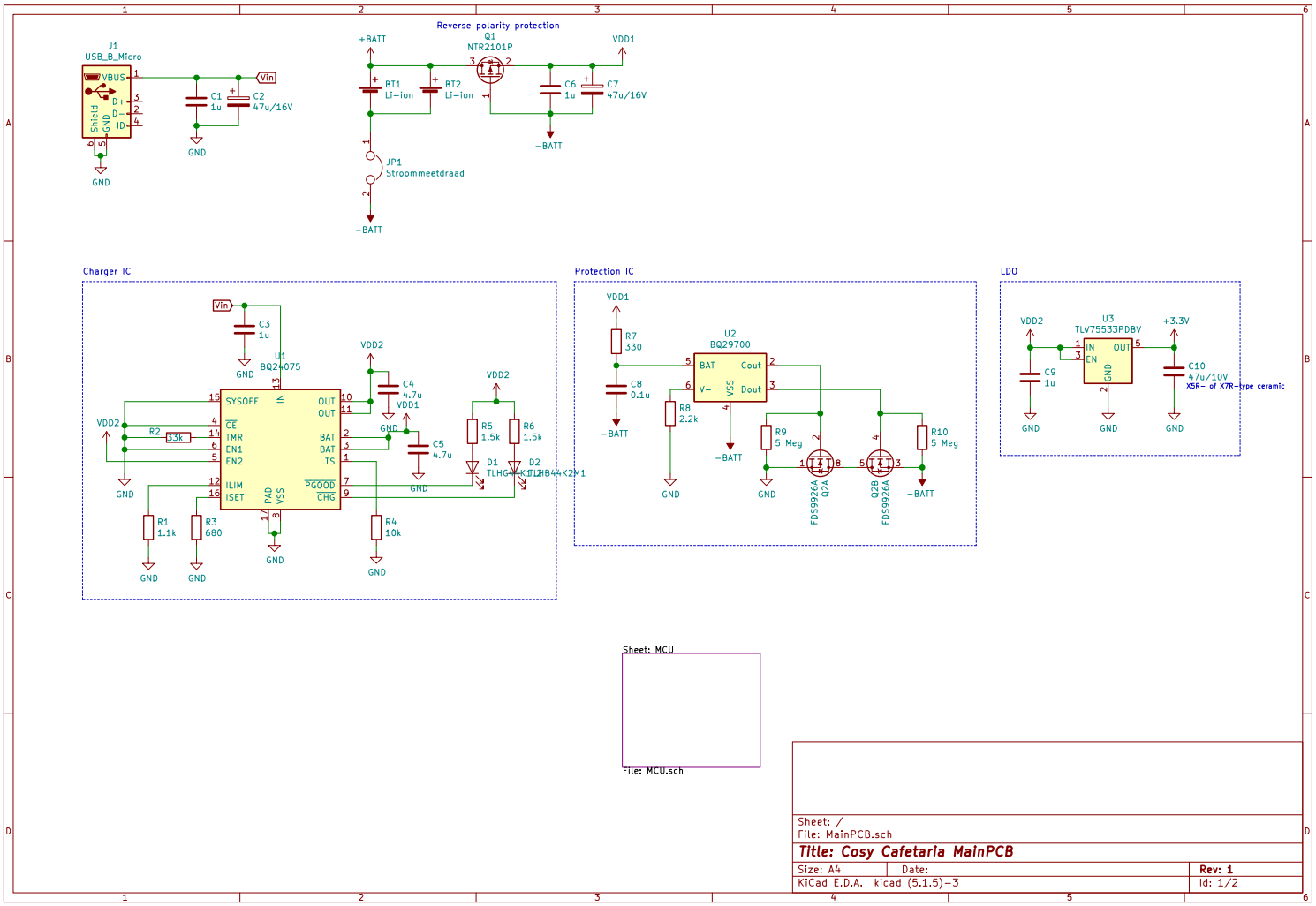
[10] "Low quiescent ldo vs. buck converter: Which is better for low power applications?" https://electronics.stackexchange.com/questions/454521/low-quiescent-ldo-vs-buck-converter-which-is-better-for-low-power-applications.

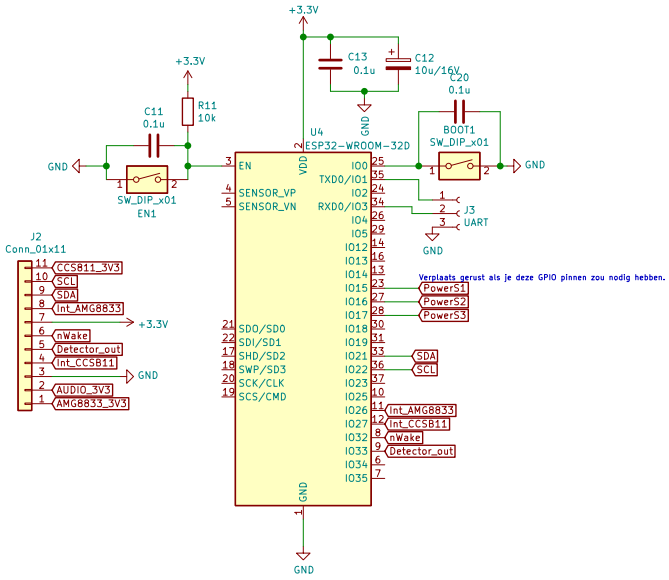
[11] *TLV755P 500-mA, Low IQ, Small Size, Low Dropout Regulator*, Texas Instruments, November 2017, datasheet.

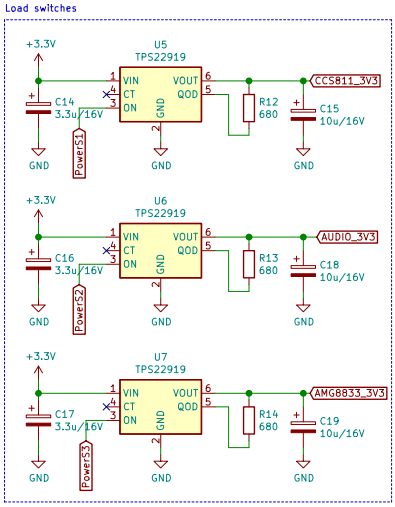
[12] L. Frenzel, "7 good reasons why you should use integrated load switches," https://www.electronicdesign.com/power-management/article/21805596/7-good-reasons-why-you-should-use-integrated-load-switches.

[13] *TPS22919 5.5 V, 1.5 A, 90-mOhm Self-Protected Load Switch with Controlled Rise Time*, Texas Instruments, October 2018, datasheet.

A MainPCB schematic

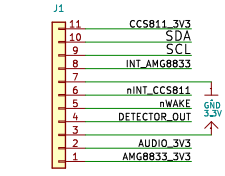

Reverse polarity protection
J1
USB_B_Micro
VBUS
D+
D-
ID
Shield
GND
C1
1u
C2
47u/16V
Vin
GND
+BATT
Q1
NTR2101P
VDD1
BT1
Li-Ion
BT2
Li-Ion
C6
1u
C7
47u/16V
−BATT
JP1
Stroommeetdraad
Charger IC
U1
BQ24075
SYSOFF
IN
OUT
CE
TMR
EN1
EN2
BAT
TS
ILIM
ISET
PGOOD
CHG
PAD
VSS
C3
1u
C4
4.7u
C5
4.7u
R1
1.1k
R2
33k
R3
680
R4
10k
R5
1.5k
R6
1.5k
D1
TLHG4K
D2
LHB4K2M1
VDD2
VDD1
Protection IC
U2
BQ29700
BAT
Cout
V−
VSS
Dout
R7
330
C8
0.1u
R8
2.2k
R9
5 Meg
R10
5 Meg
Q2A
FDS9926A
Q2B
FDS9926A
LDO
U3
TLV75533PDBV
IN
OUT
EN
GND
+3.3V
C9
1u
C10
47u/10V
X5R- of X7R-type ceramic
Sheet: MCU
File: MCU.sch
Sheet: /
File: MainPCB.sch
Title: Cosy Cafetaria MainPCB
Size: A4
Date:
Rev: 1
KiCad E.D.A. kicad (5.1.5)-3
Id: 1/2



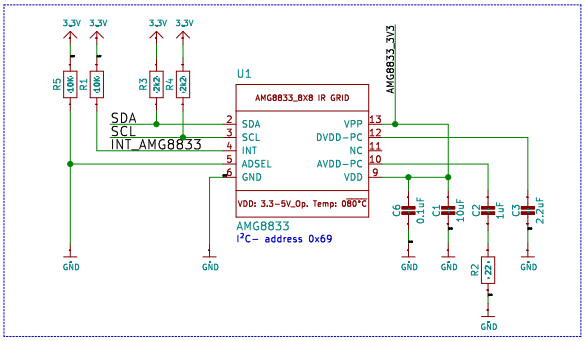


Sheet: /MCU/		
File: MCU.sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.1.5)-3		Id: 2/2

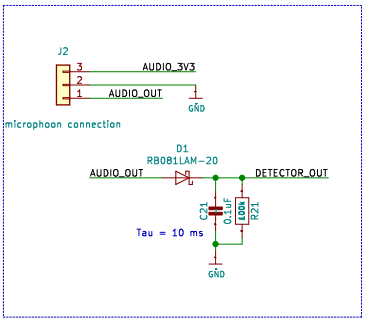
B Sensor board schematic



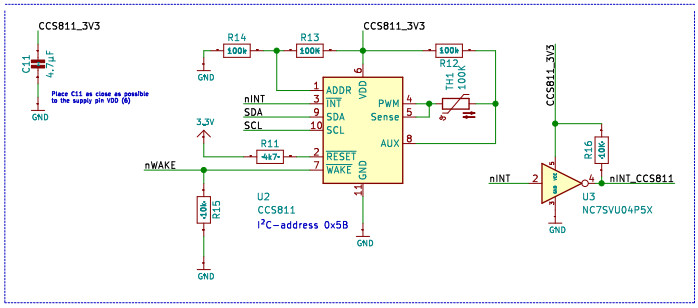
Mainboard connection



IR-sensor cricuit



Audio detector



CCS-811 circuit

Cossy Cafeteria

Sheet: /
File: SesnorBoard_ORG.sch

Title: Sensor board

Size: User	Date: 2020-02-14	**Rev: 00**
KiCad E.D.A. eeschema (5.1.5)-3		Id: 1/1