

**IT Technology**  
**Autonomous wireless charging of robot vehicles**  
**Addendum for project report**



UNIVERSITY COLLEGE LILLEBAELT

Authors  
Dainius Čeliasukas  
Stefan Kumaresu

[dain0084@edu.eal.dk](mailto:dain0084@edu.eal.dk)  
[stef1571@edu.eal.dk](mailto:stef1571@edu.eal.dk)

Thursday, June 20, 2019

## **Introduction**

This document is supplemental material for the University College Lillebaelt IT Technology Electronics project report titled “Autonomous wireless charging of robot vehicles”, done in June 3, 2019.

It describes the development process and progress done on the project after the publication of the report, and before the presentation in the final exam.

To give a short summary, the project goal was to build a system containing at least two wireless charging stations and a robot vehicle, where the vehicle is able to navigate itself to a currently unused station if it needs charging. At the time of the report hand-in, the robot vehicle was finished while the charging stations were close to completion.

The scripts and schematics previously shown in the project report have now been moved to the project’s Github repository and kept up to date:

<https://github.com/dainezasc/smart-car-charging-stations>

## Table of Contents

I.	Current and charge measurement	1
1.	Current sensor & peripherals	1
2.	Implementation	2
II.	Problems and solutions	7
1.	AC noise filtering	7
2.	Supply current and power	8
3.	MOSFET switch	8
III.	Results	10
1.	Figures	10
2.	Reflections	12
IV.	References	13
1.	Bibliography	13
2.	Useful links	13

## Table of Figures

Figure 1: The board for the ACS712 current sensor IC, which was used in the project. ....	1
Figure 2: When both electric and magnetic fields are acting on an electric charge with non-zero velocity, the resulting Lorentz force causes the charge to move in a spiraling pattern. Source: <a href="https://www.quora.com/What-is-Lorentz-force">https://www.quora.com/What-is-Lorentz-force</a> .....	1
Figure 3: The MCP3008 ADC chip. Source: <a href="https://thepihut.com/products/adafruit-mcp3008-8-channel-10-bit-adc-with-spi-interface">https://thepihut.com/products/adafruit-mcp3008-8-channel-10-bit-adc-with-spi-interface</a> .....	2
Figure 4: The sensitivity table of the MCP3008, which shows correspondence between measured voltage, output bit value and resulting amperage. Sensitivity and values were measured and calculated with the ACS712 datasheet. ....	2
Figure 5: Generic LCD screen, used in the project for displaying the input current.....	2
Figure 6: The MCP3008 circuit connected to the Raspberry Pi.....	3
Figure 7: Measured MCP3008 output, however the bit values and voltages are affected by noise and become inaccurate. ....	4
Figure 8: Working LCD display.....	5
Figure 9: Current measurement software block diagram. ....	6
Figure 10: Updated station software block diagram.....	6
Figure 11: Schematic describing the LPF placement in the circuit. ....	7
Figure 12: The updated MCP3008 circuit. ....	7
Figure 13: The current measurement script output, no longer containing fluctuations.....	8
Figure 14: The updated MOSFET switch. Notice the 1k resistor being removed from the drain side of the MOSFET. ....	9
Figure 15: Updated system hardware block diagram. ....	10
Figure 16: Finalized robot vehicle.....	10
Figure 17: Wireless charging station No. 1, with current measuring system included. ....	11
Figure 18: Wireless charging station No. 2, without current measuring system.....	11
Figure 19: Complete system of the project, with testing track included. ....	12

# I. Current and charge measurement

## 1. Current sensor & peripherals

The most significant addition to the project post hand-in is a system designed to measure electric current transferred to the robot vehicle from the charging station. The purpose of this system is to easily determine the charging efficiency and status of the station.

The system is comprised of a current sensor (in this case, an ACS712 IC), an MCP3008 ADC chip and an LCD screen for displaying measured values.

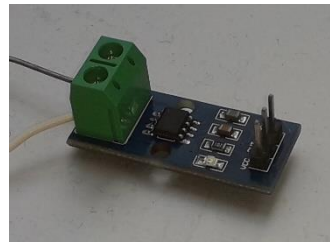


Figure 1: The board for the ACS712 current sensor IC, which was used in the project.

The ACS712 is a Hall effect based current sensor - it is activated by external magnetic fields, specifically those produced by the flow of electric current due to the Lorentz force, which is a result of combined electric and magnetic forces on a moving charge causing it to move across the conductor in a circular orbit pattern:

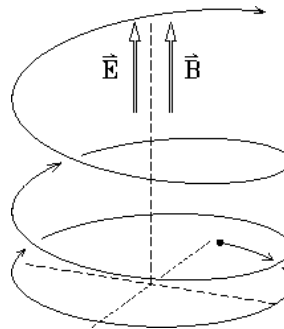


Figure 2: When both electric and magnetic fields are acting on an electric charge with non-zero velocity, the resulting Lorentz force causes the charge to move in a spiraling pattern. Source: <https://www.quora.com/What-is-Lorentz-force>

This force produces a voltage difference across a conductor, known as Hall voltage. The current sensor is able to measure the quantity of this voltage, and by extension, the electric current. This measurement is translated as an analog output value, which is directly proportional to the measured Hall voltage.

In order for the Raspberry Pi to read this value, the analog signal must first be converted into a digital signal using an analog-digital converter (ADC). For this project, the MCP3008 chip was

used. It converts the current sensor readings to a digital format, and periodically sends the measured current and voltage values to the Raspberry Pi through its SPI bus.

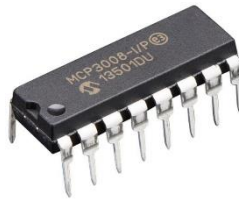


Figure 3: The MCP3008 ADC chip. Source: <https://thepihut.com/products/adafruit-mcp3008-8-channel-10-bit-adc-with-spi-interface>

BitValue	Volts	Amps
471	2,302	-3
485	2,368	-2
498	2,434	-1
512	2,5	0
526	2,566	1
539	2,632	2
553	2,698	3

Figure 4: The sensitivity table of the MCP3008, which shows correspondence between measured voltage, output bit value and resulting amperage. Sensitivity and values were measured and calculated with the ACS712 datasheet.

Lastly, the current and voltage values are then outputted to an LCD display connected to the Raspberry using the I2C bus.



Figure 5: Generic LCD screen, used in the project for displaying the input current.

## 2. Implementation

In order to do current and voltage measurements, the load resistance  $R_L$  of the current measurement system must be calculated first.

**Find the Load Resistor:**

$$V = 5V \text{ \& } I = 1A$$

$$P = V * I$$

$$P = 5V * 1A$$

$$P = 5W$$

$$R_L = P/I^2$$

$$R_L = 5W/1A^2$$

$$\underline{\underline{R_L = 5\Omega}}$$

In addition, the resolution of the MCP3008 ADC chip must be calculated. In other words, the calculation shows how many amps can be measured per bit.

**Find the Resolution:**

**ACS712 output sensitivity: 66mV/A**

$$FSR = \frac{5000mV}{1024bit}$$

$$FSR = 4,88mV/bit$$

$$A/bit = \frac{4,88mV/bit}{66mV/A}$$

$$\underline{\underline{A/bit = 73mA/bit (Resolution)}}$$

In order to interface the MCP3008 chip with the Raspberry Pi, some code that can record the MCP3008 bit values and convert them into voltage values needs to be written. Before connecting the ADC to the Raspberry, its SPI interface needs to be enabled, since the MCP3008 runs SPI and works only with this interface. This was achieved with the *Python Spidev* package (see References), which contains all the packages necessary for setting up SPI connection between the MCP3008 and the Raspberry.

After making the MCP3008 circuit on a breadboard, it was connected to the GPIO ports of the Raspberry:

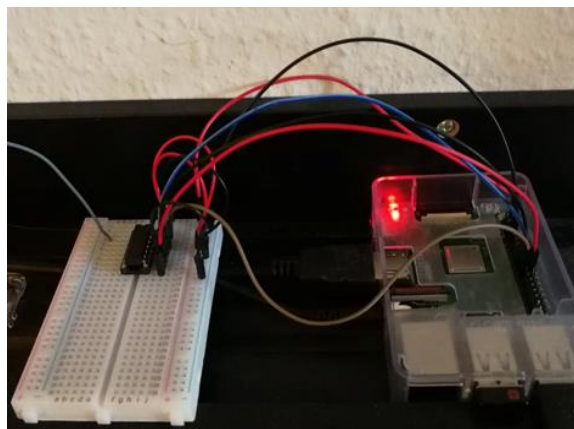
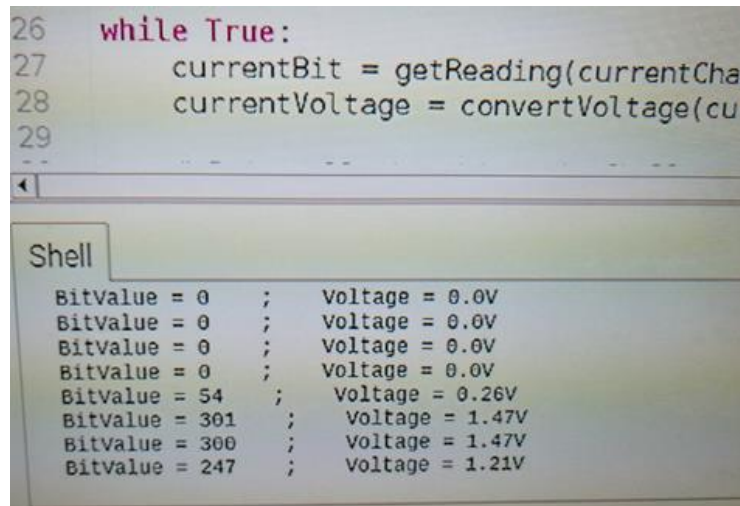


Figure 6: The MCP3008 circuit connected to the Raspberry Pi.

Before measuring current, a voltage measurement script was written and tested. It does the following:

1. Opens up the SPI bus
2. Initializes which channel the current sensor will be on
3. Pulls raw data from the MCP3008 (Bit values)
4. Converts bit values into voltage values

The initial tests of the code with the circuit were successful, however the readings of the MCP3008 are very fluctuating and imprecise at times:



The image shows a code editor with a Python script and a terminal window. The script is a while loop that reads data from the MCP3008 and converts it to voltage. The terminal shows the output of the script, which is a list of bit values and their corresponding voltage values. The output is noisy, with bit values ranging from 0 to 301 and voltage values ranging from 0.0V to 1.47V.

```
26 while True:
27     currentBit = getReading(currentChar
28     currentVoltage = convertVoltage(cur
29
--
Shell
BitValue = 0 ; Voltage = 0.0V
BitValue = 0 ; Voltage = 0.0V
BitValue = 0 ; Voltage = 0.0V
BitValue = 0 ; Voltage = 0.0V
BitValue = 54 ; Voltage = 0.26V
BitValue = 301 ; Voltage = 1.47V
BitValue = 300 ; Voltage = 1.47V
BitValue = 247 ; Voltage = 1.21V
```

Figure 7: Measured MCP3008 output, however the bit values and voltages are affected by noise and become inaccurate.

For information on how this problem was solved, see section 1 in “Problems and solutions”.

The next step was the setup of the 20x4 LCD display, which was connected to the Raspberry Pi using the I2C interface. Therefore, I2C must be enabled on the Raspberry first (to see the repository for the LCD I2C interface and driver code, see References). Afterwards, the LCD can be connected to the GPIO pins of the Pi.

The code for the LCD does as following:

1. Loads the LCD Driver
2. Prints title on first row
3. Prints bit values on second row
4. Prints offset voltages on third row
5. Prints current on the last row



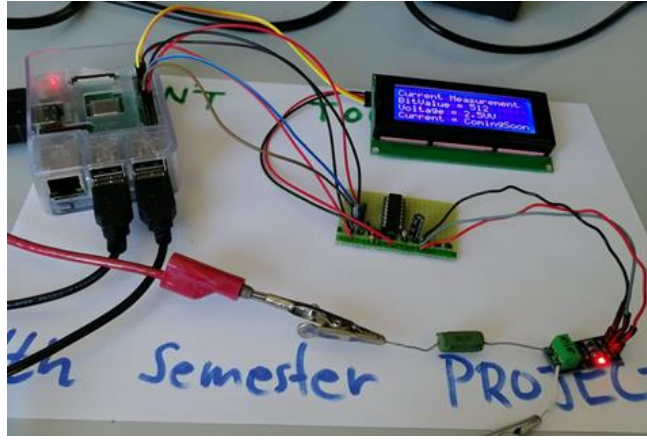


Figure 8: Working LCD display.

Once the LCD was set up, the final step is completing the code for electric current retrieval and display. These are the equations used for calculating current and offset voltage from the ADC's bit values generated by the current sensor:

$$V_{out,ACS712} = 2.5V + 0.185V \cdot I_{in}$$

$$V_{in,MCP3008} = \frac{5V}{1023bit} \cdot bitValue$$

$$V_{out,ACS712} = V_{in,MCP3008}$$

$$2.5V + 0.185V \cdot I_{in} = \frac{5V}{1023bit} \cdot bitValue$$

$$I_{in} = \frac{\frac{5V}{1023bit} \cdot bitValue - 2.5V}{0.185V}$$

$I_{in}$  is the current that passes through the transmitted coil and is displayed on the stations' LCD display.

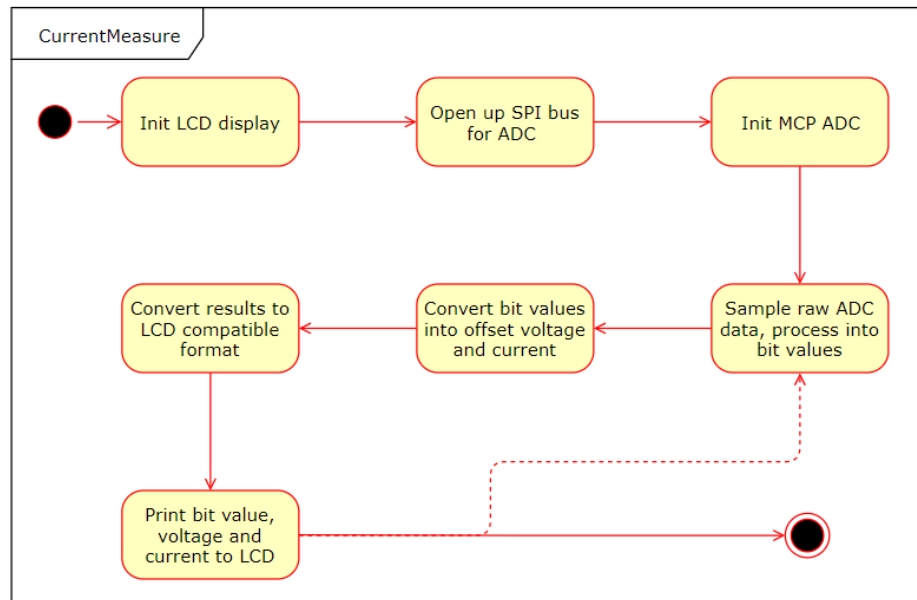


Figure 9: Current measurement software block diagram.

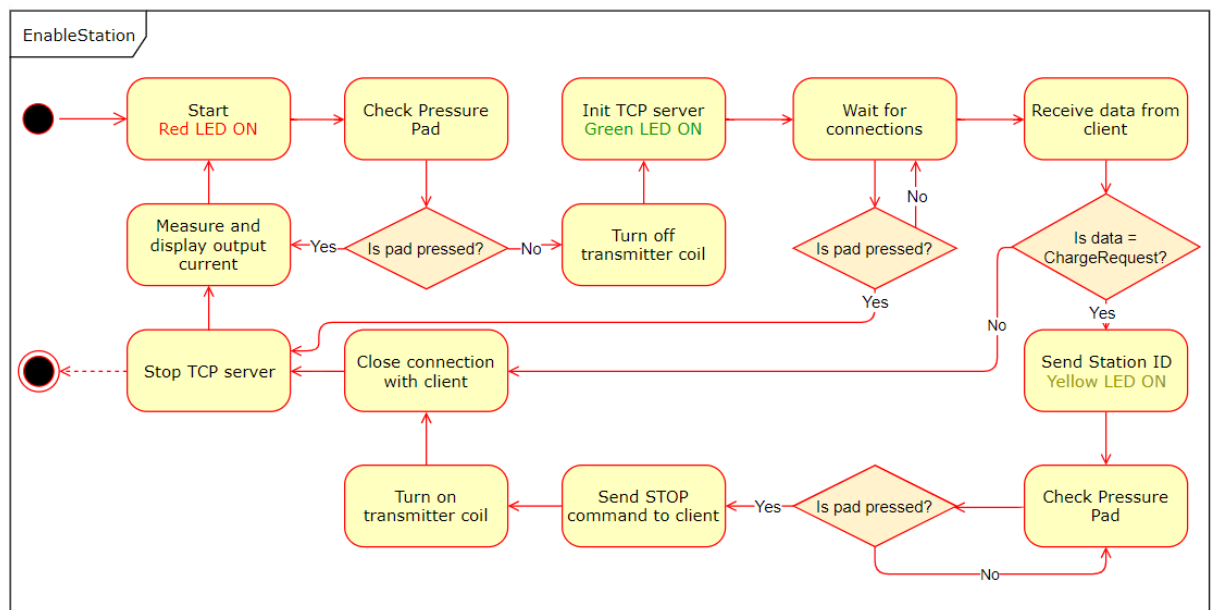


Figure 10: Updated station software block diagram.

## II. Problems and solutions

### 1. AC noise filtering

**Problem:** The readings from the MCP3008 ADC connected to the current sensor are fluctuating and inaccurate to the actual current in the circuit.

**Solution:** this problem was solved by utilizing low-pass filters (LPFs) on the MCP3008 circuit. The first LPF was placed between the signal line of the current sensor and the MCP3008s Channel 0. The second LPF was placed between the VREF on the MCP3008 and the Raspberry Pi 5V power lane.

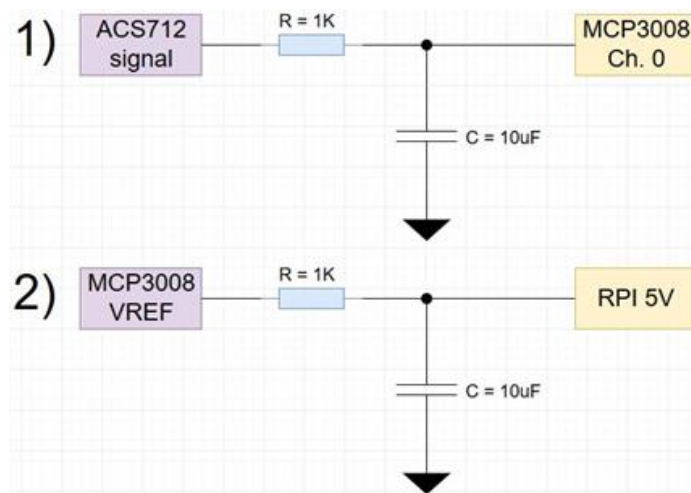


Figure 11: Schematic describing the LPF placement in the circuit.

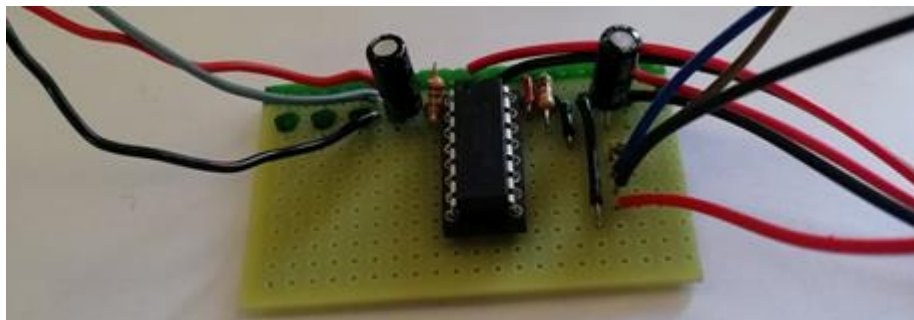


Figure 12: The updated MCP3008 circuit.

The purpose of the LPFs is to filter out high frequency noise and only let low frequency signals through, which solved the problem completely. Stable precise readings from our MCP3008 chip were finally achieved:

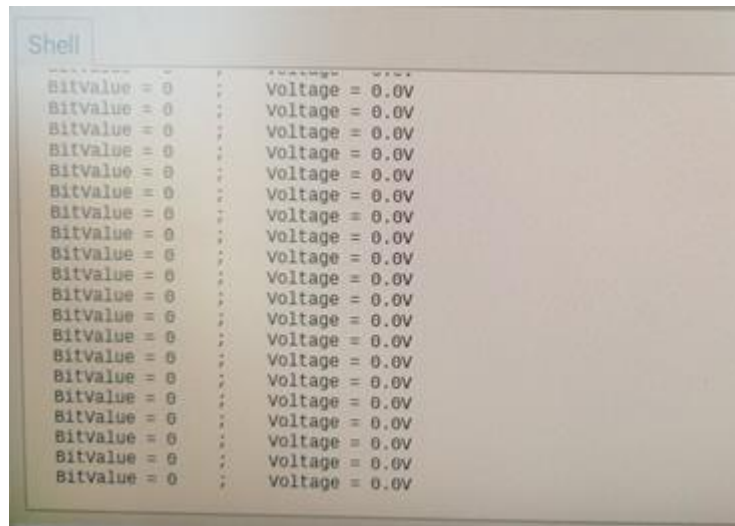


Figure 13: The current measurement script output, no longer containing fluctuations.

## 2. Supply current and power

**Problem:** The Raspberry Pi does not receive the required 5V from the PSU when it was connected to the coil, MOSFET switch and LED board. As a result, any change in current (toggling the LEDs or the coil, for example) in the circuit caused the Pi to shut down and restart.

**Solution:** The cause of this problem was determined to be the circuit needing more current than the 5V/1A PSU could provide. Combined, the Raspberry and the coil and other peripherals consumed about 2.5A, much more than the previous PSU could give. The solution, therefore, was to simply swap out the PSU with one that can supply more current, about 2.5-3A. For this project, both stations were set up with a pair of PSUs, one for the Raspberry (5V/2A) and one for the coil (5V/1A) due to lack of available 5V/3A PSUs.

## 3. MOSFET switch

**Problem:** The MOSFET switch works, but the 5V/1A AC/DC wall socket power supply is unable to supply both the Raspberry and the transmitter coil in the charging station. The transmitter coil is only getting ~3.6 V after it has been routed through the MOSFET, which is not enough to supply the powerbank connected to the robot vehicle.

**Solution:** this problem was mentioned in the main project report and was left unsolved at the time. Through testing, the cause of the voltage drop was determined to be the 1k ohm resistor connected to the drain side of the MOSFET switch, which is usually placed in circuits with running currents up to 10 A and much higher voltages. This circuit is comparatively low voltage and current, and the wireless charging modules are designed to be plugged straight to 5V and GND rails, therefore the resistor is not necessary.

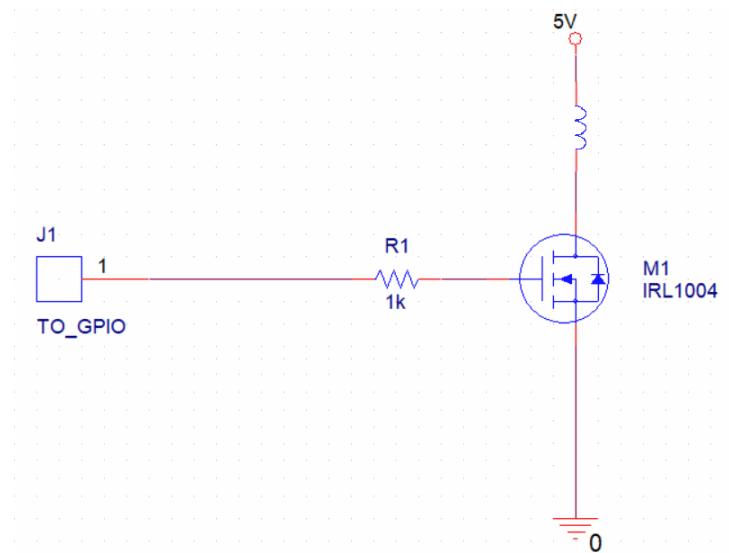


Figure 14: The updated MOSFET switch. Notice the 1k resistor being removed from the drain side of the MOSFET.

In addition, the MOSFETS have been fitted with heatsinks to dissipate excess heat more efficiently.

### III. Results

#### 1. Figures

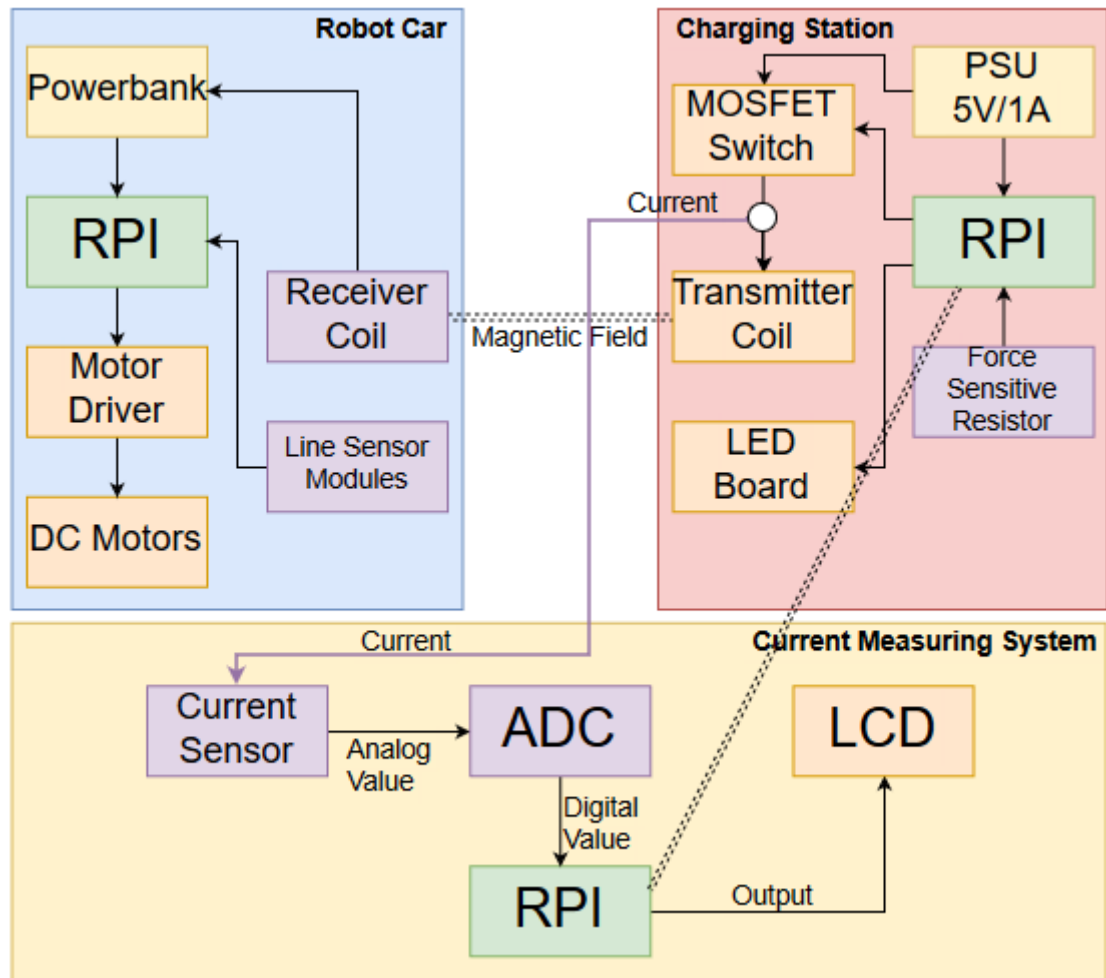


Figure 15: Updated system hardware block diagram.

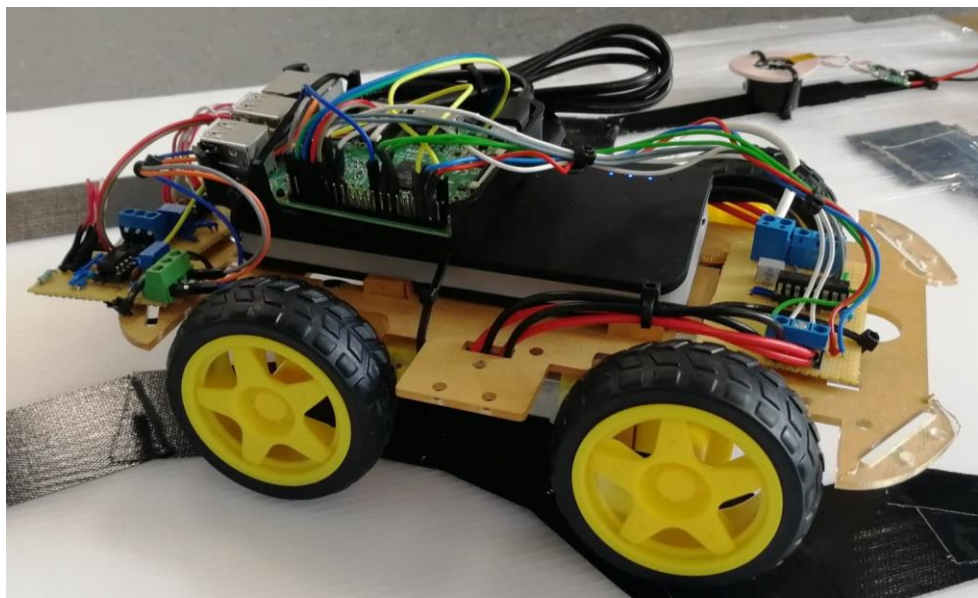


Figure 16: Finalized robot vehicle.



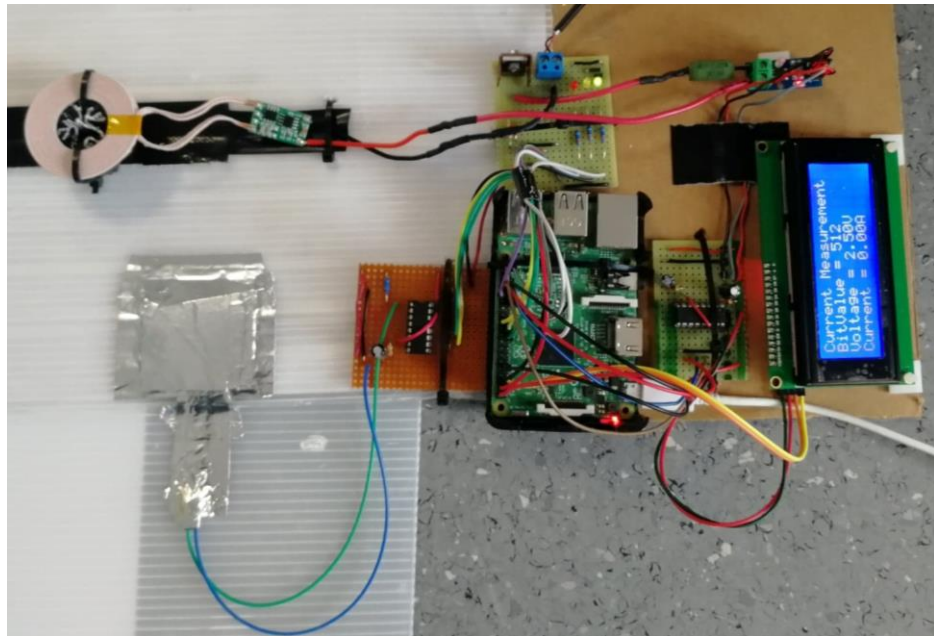


Figure 17: Wireless charging station No. 1, with current measuring system included.

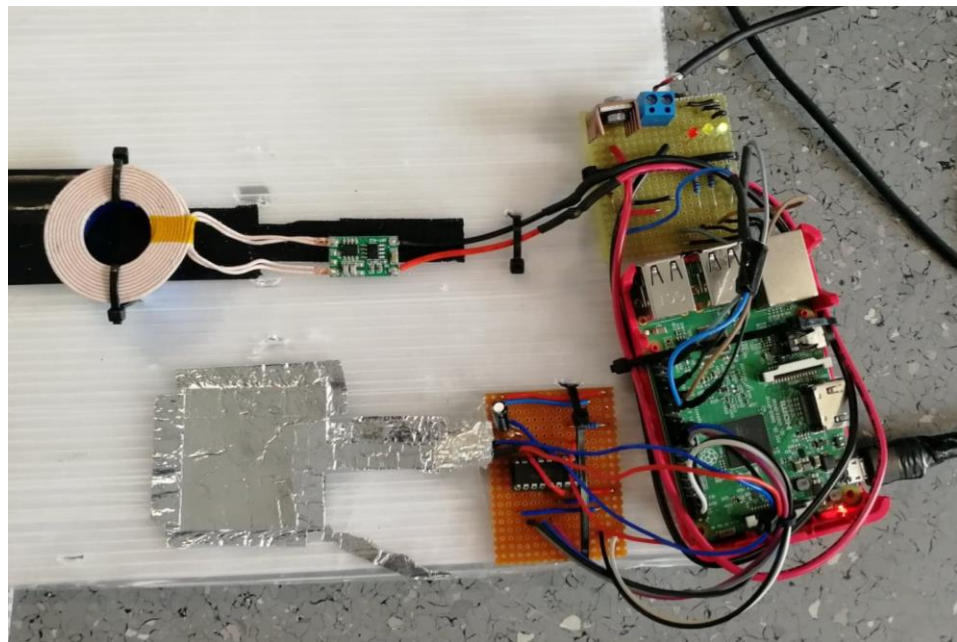
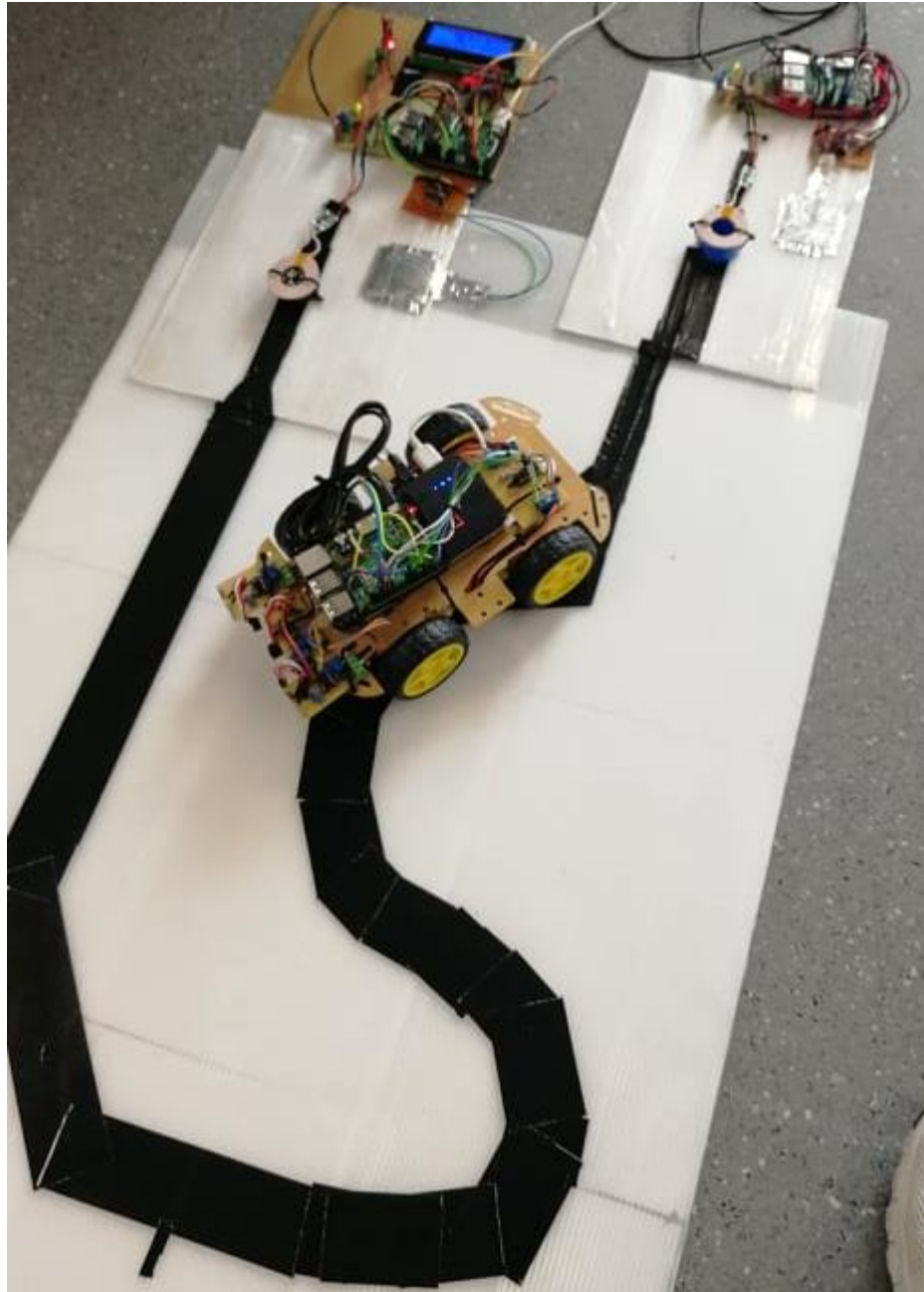


Figure 18: Wireless charging station No. 2, without current measuring system.



*Figure 19: Complete system of the project, with testing track included.*

## **2. Reflections**

For a project with a development time this short, the results, in the author's opinion, are impressive. Even though the delivery of the vehicle to the charging station is not consistently successful, the project as a whole can be deemed successful, at least theoretically.

In addition, as it was intended from the start, the resulting system can serve as a good base for improvements and further optimizations by next generations of IT Technology students.



## IV. References

### 1. Bibliography

Sparkfun Electronics. “ACS712 - Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor”. Retrieved from <https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf> at June 17, 2019.

PowerStream Technology. “How to calculate battery run-time.” Retrieved from <https://www.powerstream.com/battery-capacity-calculations.htm> at June 17, 2019.

### 2. Useful links

Python Spidev package repository:  
<https://github.com/Gadgetoid/py-spidev>

LCD-Raspberry interface code repository:  
<https://github.com/the-raspberry-pi-guy/lcd>