RoboCup Junior
2019
Soccer Lightweight
S.P.Q.R. Team
I.T.I.S. Galileo Galilei,
Rome

Technical Documentation

Team Members

Dario Casagrande

Age: 19

Hobbies: photography e HI-FI enthusiast

Role: Hardware engineer, PCB designer

Emanuele Coletta

Age: 16

Hobbies: Reading & game developing

Role: Software engineer

Emanuele Latino

Age: 18

Hobbies: Film photography & music

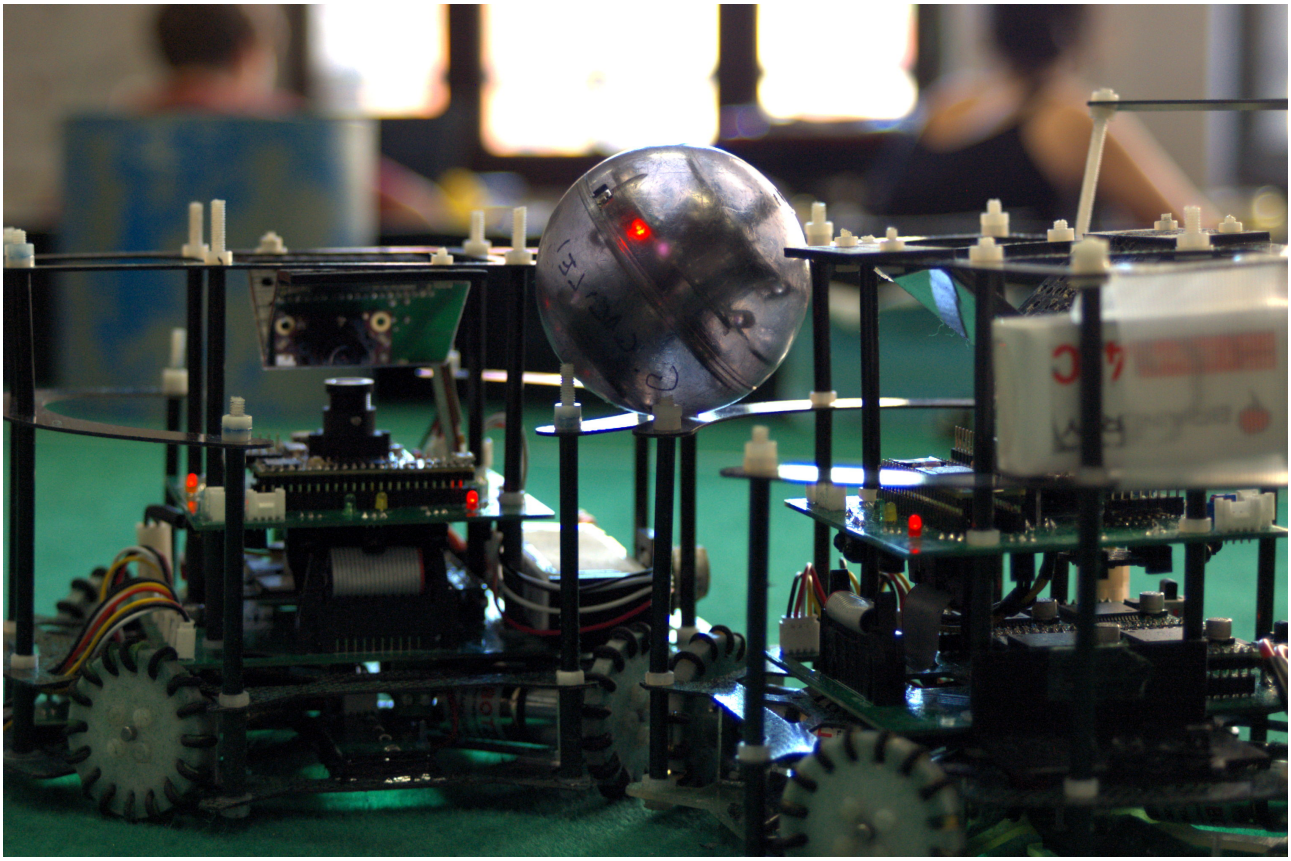
Role: Software engineer

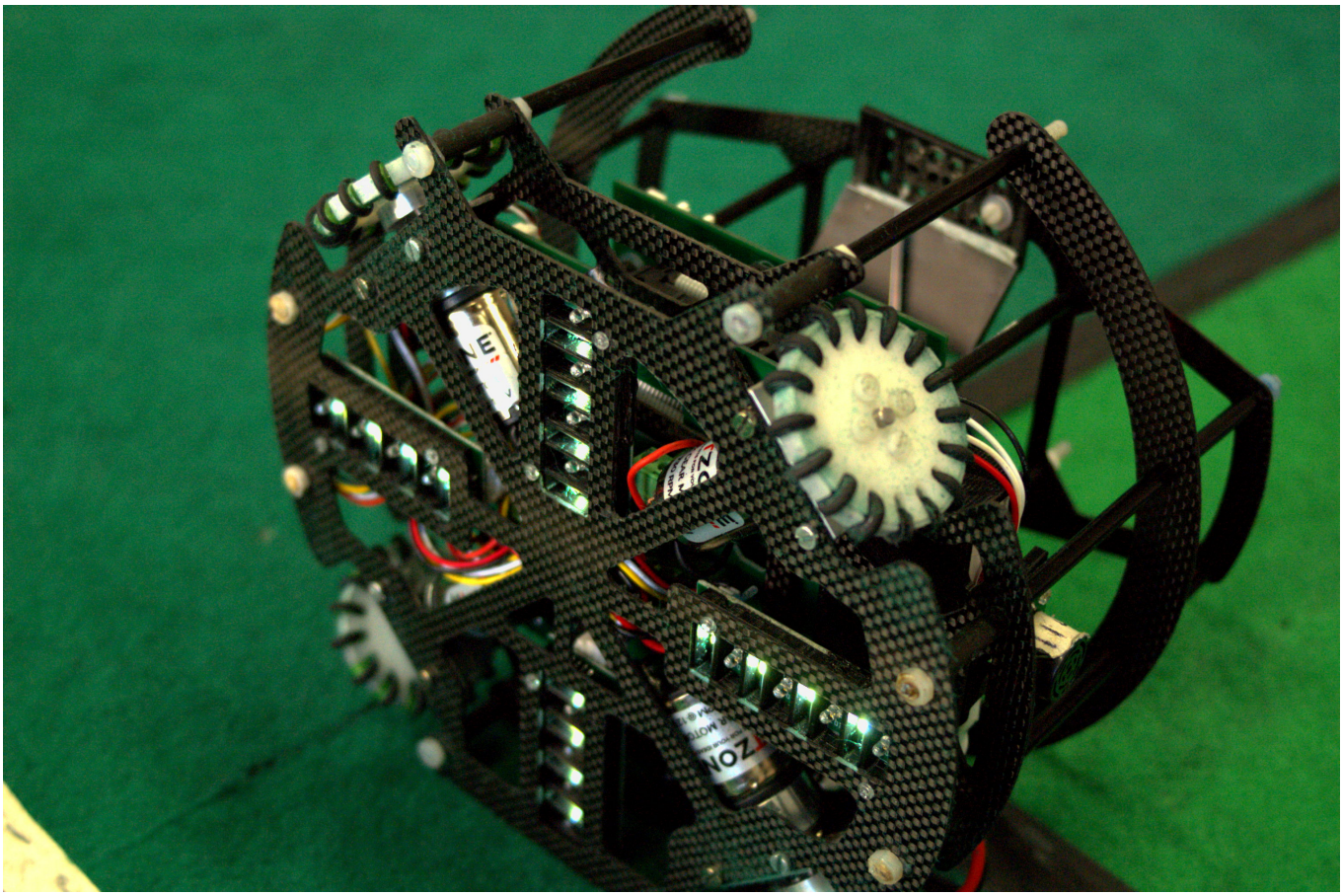
Achille Ghezzi

Age: 19

Hobbies: Electronics & climbing

Role: Mechanical engineer, designer





FURIO

Furio is two months old. He is primarily a goalie, but he can take part as a keeper as well because the role is decided by a switch. He mounts a BT module to communicate with his teammate and an IMU to know where he is. New PCBs and a new design were crafted for him, so now we can fully use the implemented camera, and we can use four motors instead of three.

The software was improved, we have new strategies and a better control over his movements.

CORNELIA

Cornelia is two months old, and she is our keeper. , but she can take part as a keeper as well because the role is decided by a switch. She mounts a BT module to communicate with his teammate and an IMU to know where he is. New PCBs and a new design were crafted for her, so now we can fully use the implemented camera, and we can use four motors instead of three.

The software was improved, we have new strategies and a better control over her movements.

THE NEW GENERATION

Our robot works with two boards:

- One for the microcontroller (Teensy 3.5) and the ball sensors
- One for the motors and the line sensors

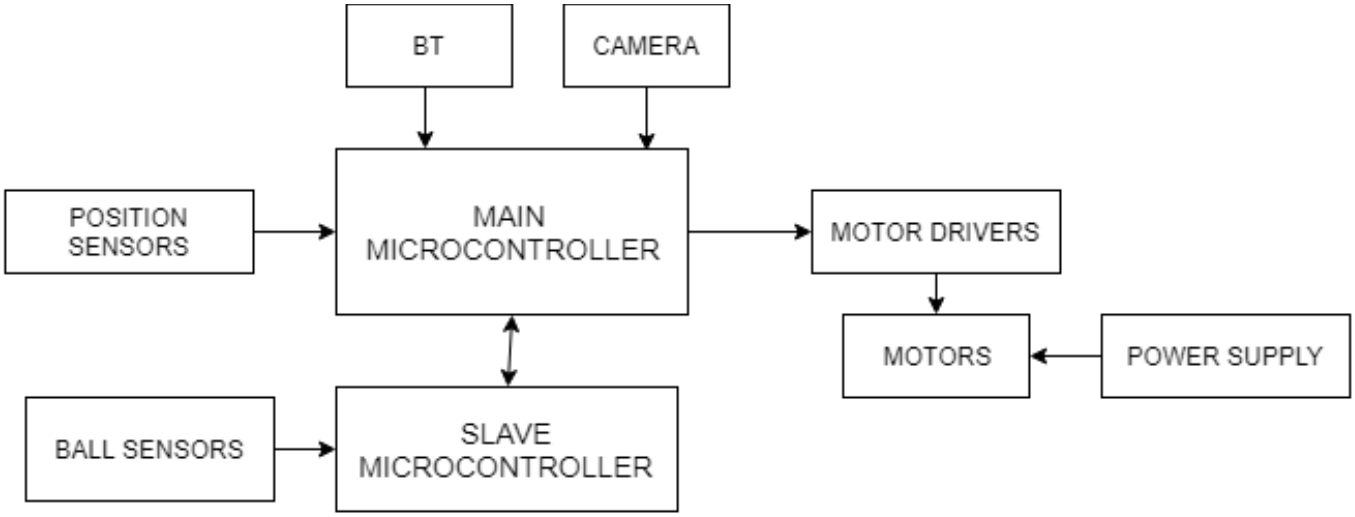
The design was made with Eagle CAD, Fusion and AutoCAD.
Many improvements were made: the hardware is now fully funcional and more practical, as the design is now much more efficient and elegant than before. We now have:

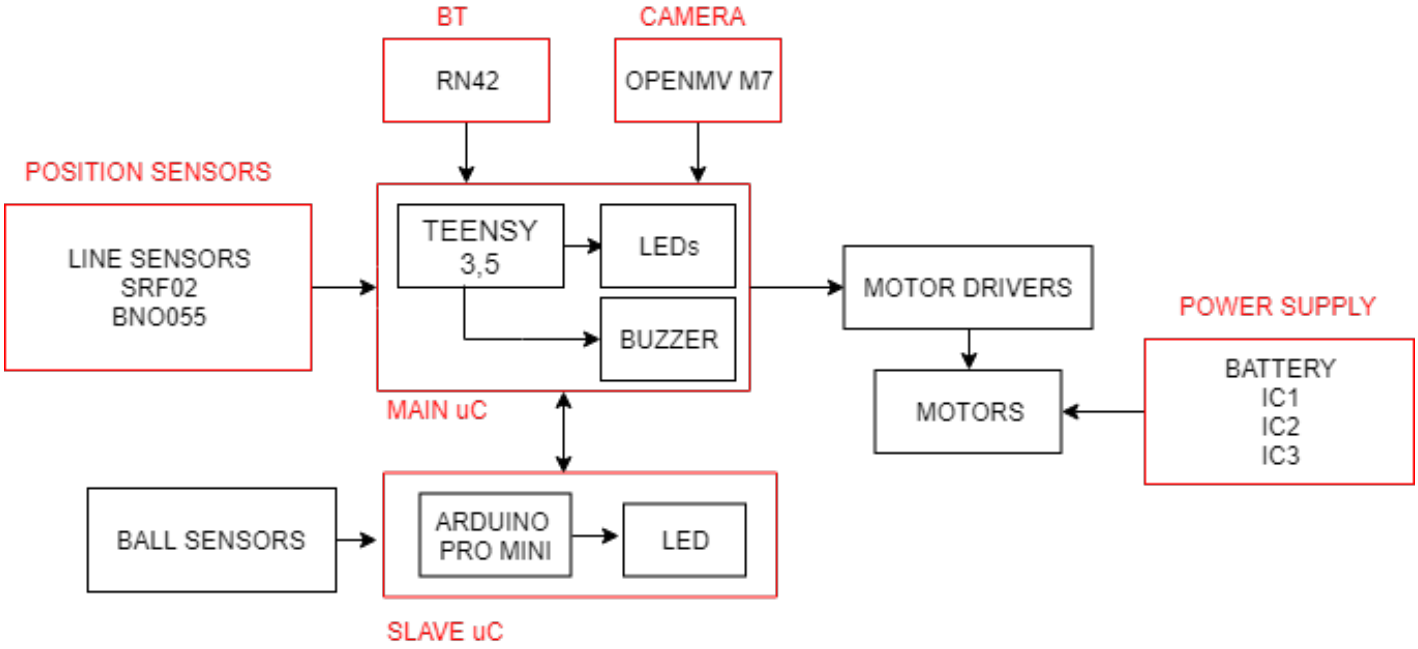
- a new camera with a mirror
- smaller PCBs, easier to repair and to carry
- 16 ball sensors instead of 20, easy to manage and precise
- 4 motors to move more smoothly and to have more control over the movements
- new line sensors
- new carbon fiber structure and design
- Teensy 3.5 and Arduino Pro Mini
- Connectors between the boards on the bottom, for a linear design

From the software's side, we now have:

- a totally new management of the ball
- a new management of the robot's movement
- a new system to keep up with the robot's zone
- a new IDE, from Arduino IDE to VSCode
- a more evolved way to act

ROBOT BLOCK DIAGRAM



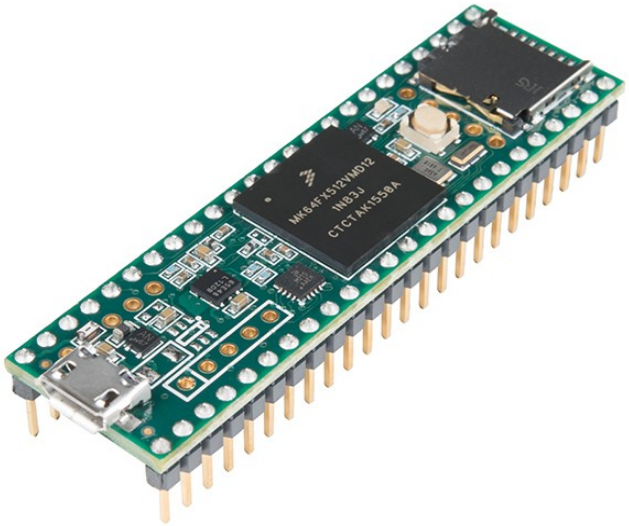


WHAT'S ON BOARD?

MICROCONTROLLERS

- Teensy 3.5

A lightweight, yet powerful microcontroller to work with and compatible with the Arduino Libraries. He is the “master”, so it commands a secondary board, to cooperate. It's the robot's mind, it decides what to do based on the detected inputs, and it' s really fast!



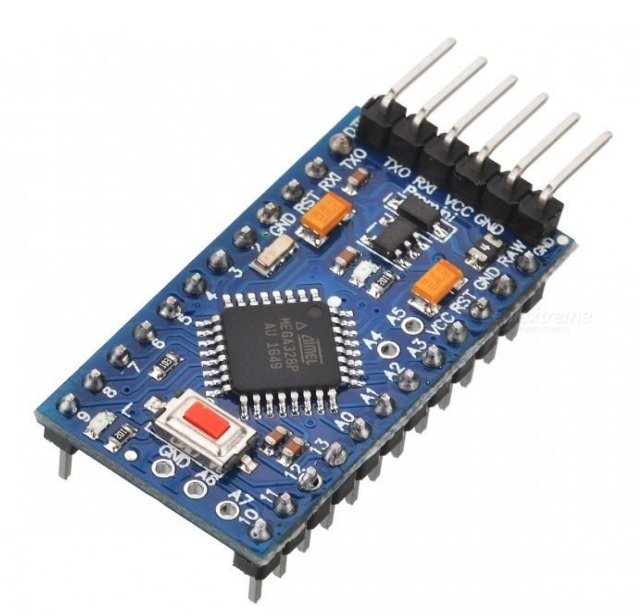
It also supports multithreading and it has its own library for it, unfortunately, it's not official, therefore, it's not stable. Our Teensy mounts a Cortex M4 MK64FX512VMD12 with floating point unit, it can go up to a clock of 180 Mhz! It has 256 KB RAM and 4K EEPROM.

Communication wise it supports SPI, it has 3 ports, and one of them is with FIFO, as well as 3 I2C ports and 6 Serial Ports with Fast Baud Rate and FIFO.

Every digital I/O pin has a 5 V tolerance and an Interrupt capability, that makes it very versatile, because there are 57 of them. Every pin is also multifunctional, there are 20 PWM pins, 25 Analog Output pins and 2 Analog Input pins.

More I/O pins are available at small surface mount pads on the back side. The 6th serial port and 3rd SPI port are on these pins. They're not as easy to access as the main 42 through-hole pins on the outside edge, so we don't use them.

- Arduino Pro Mini



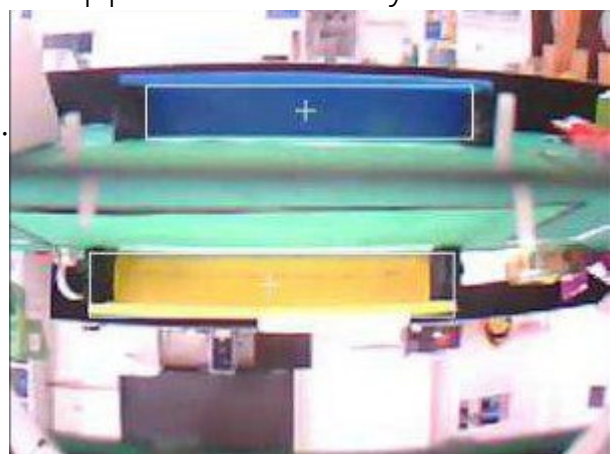
The Arduino Pro Mini is a microcontroller board based on the Atmega328P.
We use it as a "slave" microcontroller, it manages the ball reading, interpolating the sensors' data and sending it back to the master via SPI.
It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, an on-board resonator, a reset button, and holes for mounting pin headers. A six pin header can be connected to an FTDI cable or Sparkfun breakout board to provide USB power and communication to the board.
The Arduino Pro Mini is intended for semi-permanent installation in objects or exhibitions. The board comes without pre-mounted headers, allowing the use of various types of connectors or direct soldering of wires. The pin layout is compatible with the Arduino Mini.
There are two version of the Pro Mini. One runs at 3.3V and 8 MHz, the other at 5V and 16 MHz.
The Arduino Pro Mini was designed and is manufactured by SparkFun Electronics.

CAMERA

- OpenMV M7

The OpenMV M7 Cam is a very powerful, valuable and affordable camera. We mount one of them on each robot, with a "V" shaped mirror to see the entire camp and to track the goalposts by color.

It's based on the STM32F765VI ARM Cortex M7 processor running at 216 MHz with 512KB of RAM and 2 MB of flash, it also features a floating point unit (FPU) which supports Arm® double-precision and single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) which enhances the application security. This makes it mount a µSD Card socket for video recording, SPI, Serial and I2C ports. All I/O pins have Interrupt capability and PWM, their output is 3.3V and 5V tolerant. This camera is also really flexible for all needs, with its OV7725 image sensor, it's capable of taking 640x480 8-bit Grayscale images or 640x480 16-bit RGB565 images at 60 FPS when the resolution is above 320x240 and 120 FPS when it is below. The lens is unmountable and it can be changed with other ones.



BLUETOOTH

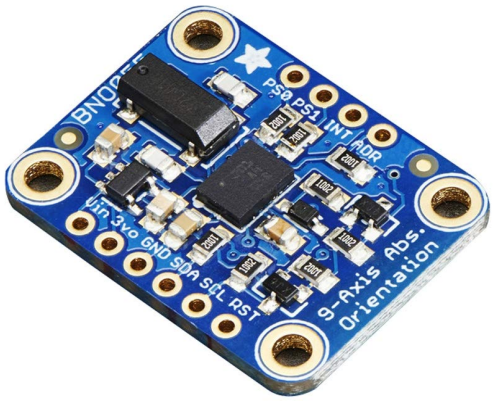
- RN42 Mate Silver

These Bluetooth modems work as a serial (RX/TX) pipe, and are a great wireless replacement for serial cables. Any serial stream from 2400 to 115200bps can be passed seamlessly from the computer to the target. It has the same pin out as the FTDI Basic, and is meant to plug directly into an Arduino Pro, Pro Mini, or LilyPad Mainboard. To make in work with our Teensy 3.5, we swap TX and RX. The RN-42 is perfect for short range, battery powered applications, it uses only 26uA in sleep mode while still being discoverable and connectable. The Bluetooth has on-board voltage regulators, so it can be powered from any 3.3 to 6VDC power supply. It also features:

- v6.15 Firmware
- Hardy frequency hopping scheme - operates in harsh RF environments like WiFi, 802.11g, and Zigbee
- Encrypted connection
- Frequency: 2.402~2.480 Ghz
- Serial communications: 2400-115200bps
- Operating Temperature: -40 ~ +70C
- Built-in antenna

POSITION SENSORS

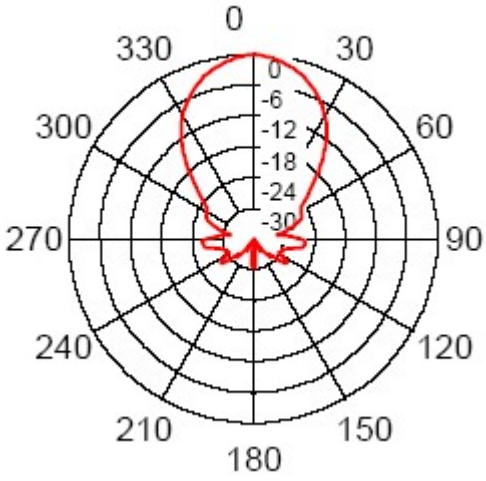
- AdaFruit BNO055



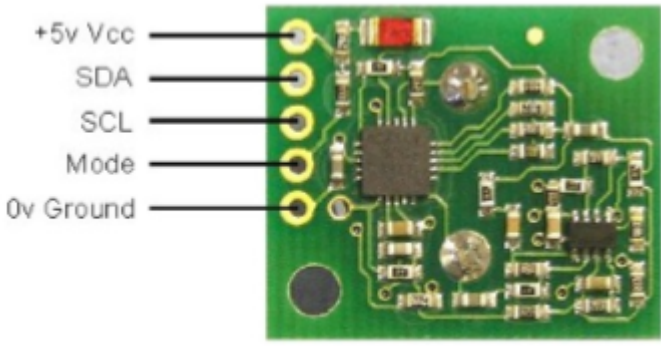
The BNO055 is an intelligent 9-axis absolute sensor. It has an embedded Cortex M0 ARM processor to perform the 9-axis sensor fusion, as well as an accelerometer, a gyroscope and a magnetometer to orient itself. No external magnetometer and no microcontroller
processing is required; again the quaternions, linear acceleration, gravity vector, and heading information are directly readable from the BNO-055 registers. This is a compact and powerful motion sensing solution that makes absolute orientation and sophisticated motion control available also for an Arduino board.
This small-form-factor board is hardwired for I2C communication with 4K7 pull-up resistors on the board.
The hardware sensor fusion is updated at a fixed frequency = 100 Hz i.e. measures are performed every 10 ms.
Our software programs the BNO-055 in Mode IMU-PLUS, and it's mounted upside-down to save some PCB space, so we use the P7 configuration. We have a way to control it through a command line interface created by us.

- Ultrasonic range finder SRF02

We use four SRF04 US sensor to detect the walls, the various robots and to locate ourselves in a preassigned zoneIndex via code. One part of the sensor is a speaker that sends out a sound wave. The other part is a microphone that then measures how long it takes for the sound wave to come back. The longer it takes to come back, the further away the object. After a long evaluation of the range finders on the market we decided to measure the robot position in the playground using SRF02 Ultrasonic range finder by Devantech connected via I2C protocol. Each sensor can be identified with an address chosen in a pool of 16; so it is theoretically possible to connect 16 sensors on a single robot.

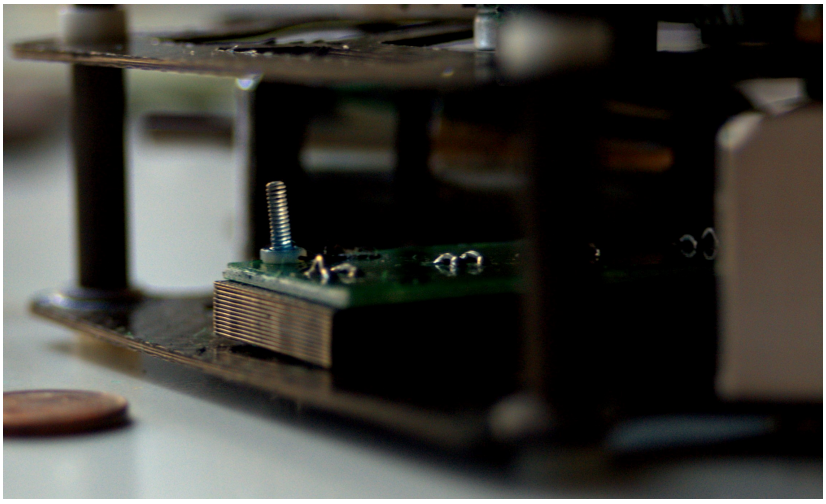


Our robots are equipped with 4 sensors pointed to North, South, East and West.



- Line Sensors

Our line sensors are brand new: we created them from scratch to be more efficient than others sold on the market. They include:



- Four white LEDs
- Four phototransistors
- Four capacitors
- Four resistors

These line sensors are analog, which means that we can decide how to manage them. We decided to put them in a totally new layout, 90° apart from each other and between the motors to decrease the Out Of Bounds percentage of our robots.

- Ball Sensors

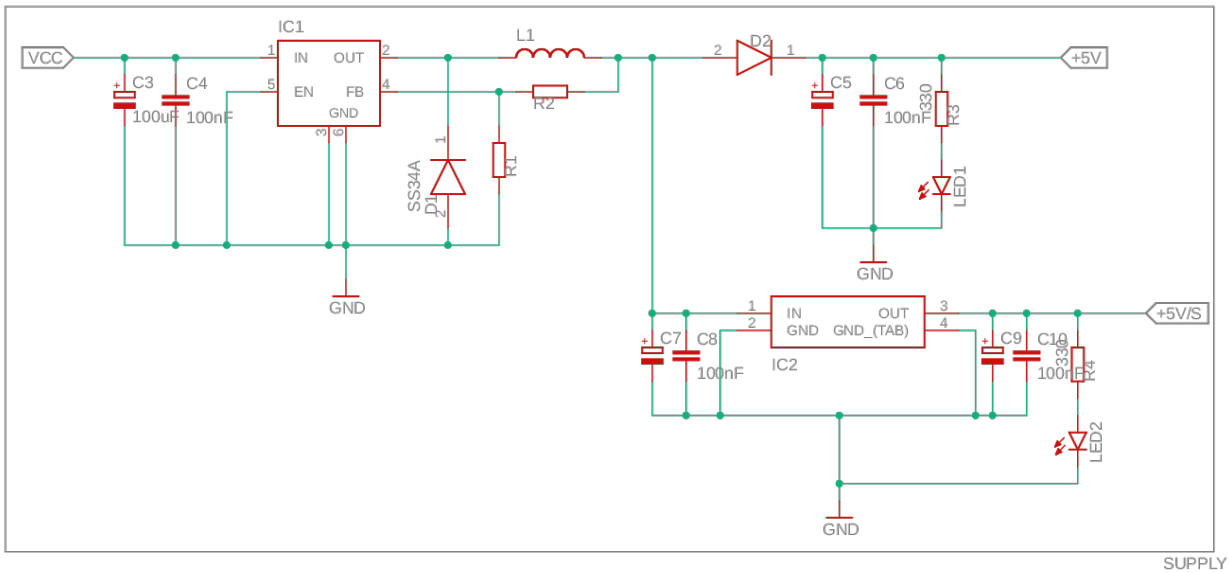
We use IR sensors to find the ball in the field, so we opted for 16 Vishay TSSP4038. They are divided in 4 groups, each group with a RC filter to make the supply more stable.

These sensors are less triggerable than the ones we used last year, so we can avoid mounting pipes for each one because the reading is already precise enough.

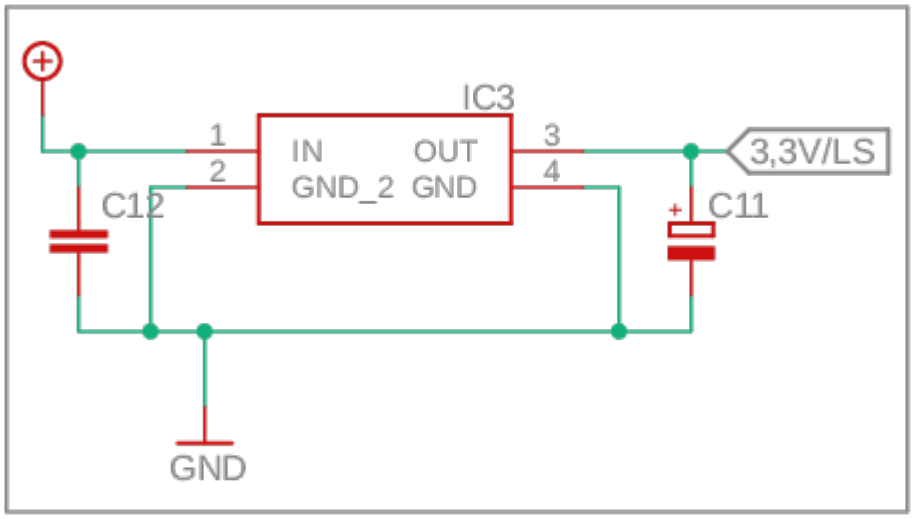
POWER SUPPLY

This new robot generation's power supply is very different from the past. The main power supply is still the same 3 cell, 12V LiPo battery, but it's paired with different components for a higher efficiency. First of all, every part of the power supply, that includes battery and voltage regulators, is on the motor board, and it's then transferred to the uC board with a Molex connector. Apart from motors, which are rated for 12V, everything needs a lower tension. We use a LM2596 buck converter to bring the voltage down to 5.7V. After this the supply is divided in two branches; we use a Schottky diode (0.7V drop) to achieve unstable 5V to make our microcontrollers, camera, IMU and bluetooth (all of them have a on-board 3,3V stabilizer). For components that need a stable supply (ball and distance sensors), we have a LDO 5V linear stabilizer.

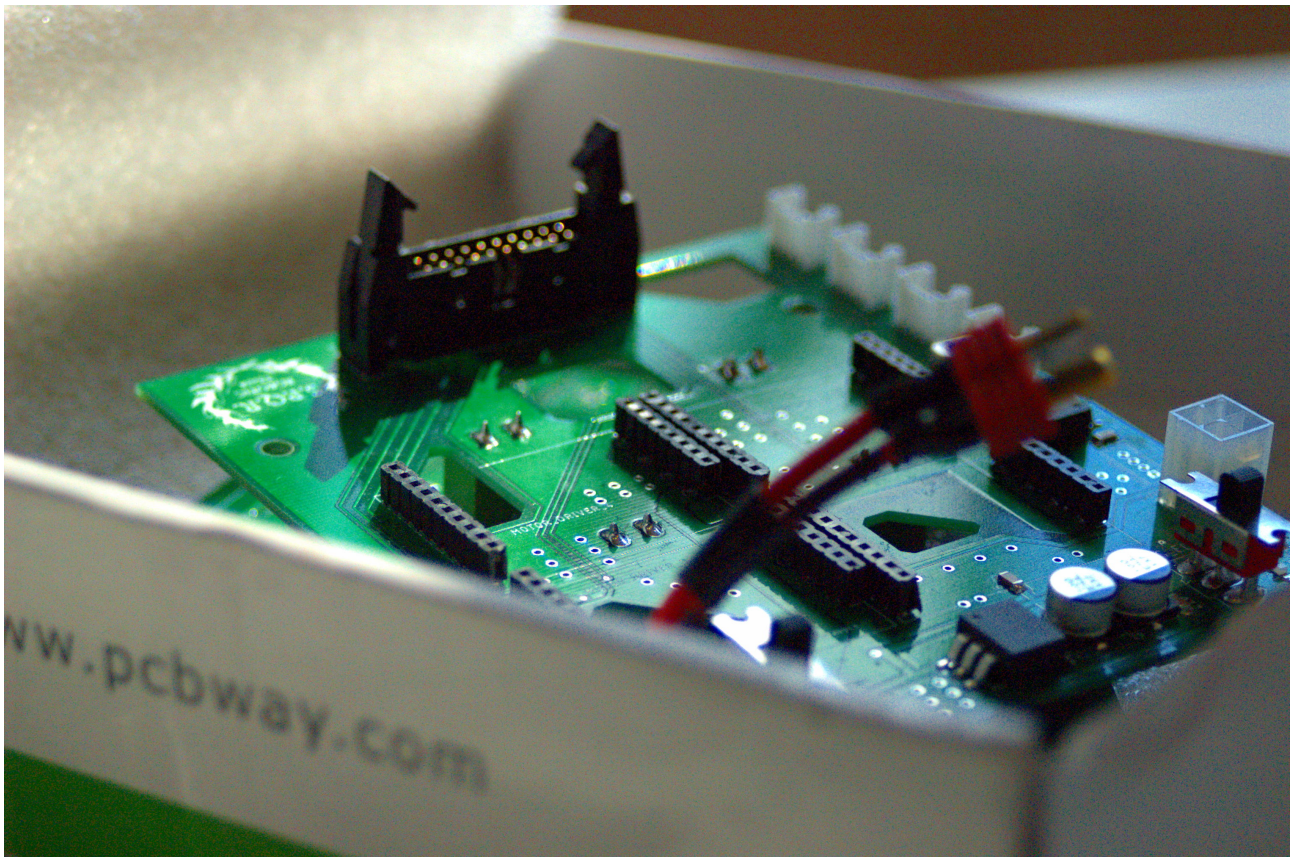
Since our microcontroller reads analog values up to 3.3V, we supply our line sensors 3.3V with a LDO linear stabilizer. This way we have a good reading and we don't risk breaking the Teensy.



Both the supply lines have a power-on LED which shuts down when there's a short circuit, so the first thing we do when we turn on the robot is looking at them.



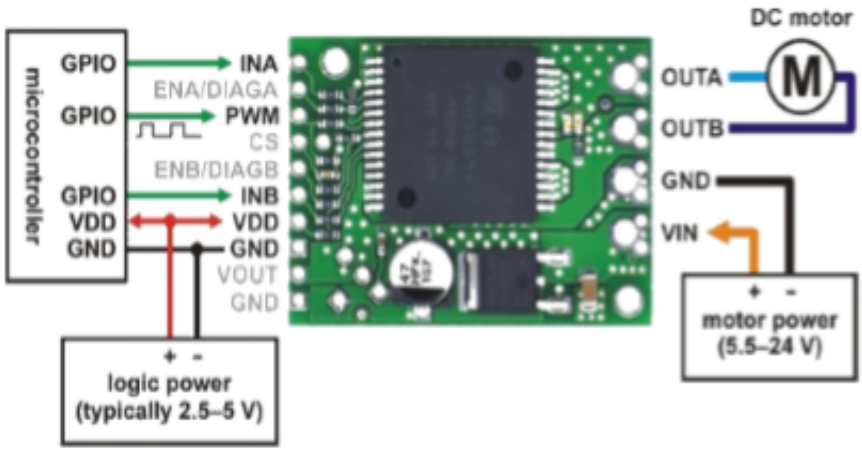
LINE_SENSORS_SUPPLY



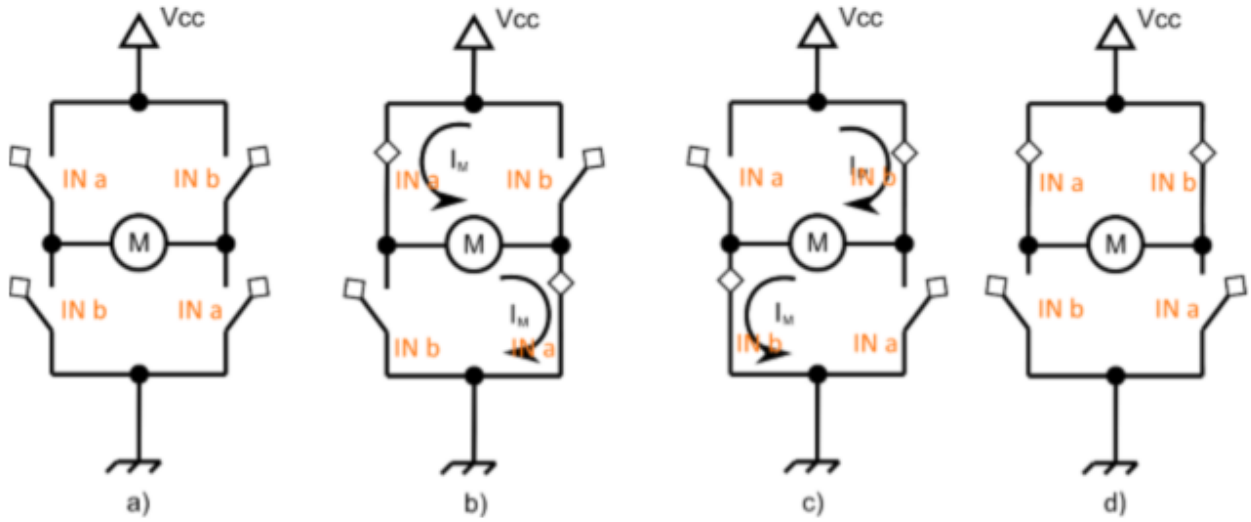
DRIVERS & MOTORS

- VNH5019 Motor Driver

To control the four motors, we use the VNH5019 Motor Drivers by Pololu. They're compact, easy to use and powerful, we mount four of them on our motor board. This module is a compact breakout board for ST’ s high-power VNH5019 motor driver IC, a fully integrated H-bridge that can be used for bidirectional speed control of a single brushed DC motor.The board incorporates most of the components of the typical application diagram on page 14 of the VNH5019 datasheet, including pull-up and current-limiting resistors and a FET for reverse battery protection. It ships fully populated with its SMD components, including the VNH5019, as shown in the product picture.



Type	$R_{DS(on)}$	I_{out}	V_{ccmax}
VNH5019A-E	18 mΩ typ (per leg)	30 A	41 V



Features:

- Operating voltage: 5.5 – 24 V1
- Output current: 12 A continuous (30 Peak)
- 3V-compatible inputs
- PWM operation up to 20 kHz, which is ultrasonic and allows for quieter motor operation
- Current sense output proportional to motor current (approx. 140 mV/A; only active while H-bridge is driving)
- Motor indicator LEDs (indicates what the outputs are doing even when no motor is connected)

Robust:

- Reverse-voltage protection to -16 V
- Can survive input voltages up to 41 V
- Undervoltage and overvoltage shutdown
- High-side and low-side thermal shutdown
- Short-to-ground and short-to-Vcc protection

The following table shows the influence of INA INB status on the operative mode, the pin OUTA (Output of half-bridge A connected to one terminal of the DC motor). and the pin OUTB (Output of half-bridge B connected the other terminal of the DC motor).

Case	IN_A	IN_B	OUT_A	OUT_B	CS (V_{CSD} = 0 V)	Operating mode
d)	1	1	H	H	High imp.	Brake to V_{CC}
c)	1	0	H	L	$I_{SENSE} = I_{OUT}/K$	Clockwise (CW)
b)	0	1	L	H	$I_{SENSE} = I_{OUT}/K$	Counterclockwise (CCW)
a)	0	0	L	L	High imp.	Brake to GND

The CS pin is the Current sense output. If connected to a resistive load it is possible to check the wheel overload. This pin is not yet implemented In our robots. The module needs 2 different power supply sources:

- VIN from the battery (12 V in our robot) for the motors
- VDD (5V) for the digital components

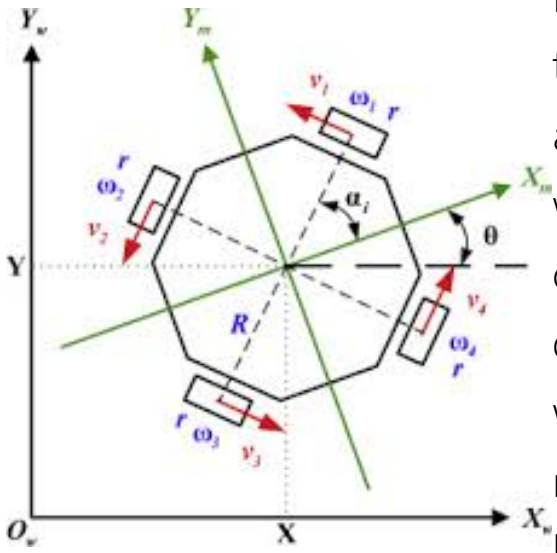
- Motors

Our motors are produced by RobotZone, they're cylindrical brushed DC motors. The gearmotors all have the same 25 mm diameter case and 4 mm diameter gearbox output shaft, so it is generally easy to swap one version for another if our design requirements . Stalling or overloading gearmotors can greatly decrease their lifetimes and even result in immediate damage. For these gearboxes, the recommended upper limit for instantaneous torque is 200 oz-in (15 kg-cm. Stalls can also result in rapid (potentially on the order of seconds) thermal damage to the motor windings and brushes, especially for the versions that use high-power (HP) motors; a general recommendation for brushed DC motor operation is 25% or less of the stall current. This robot generation uses 4 motors.

Voltage (Nominal)	12V
Voltage Range (Recommended)	3V - 12V
Speed (No Load)*	730 rpm
Current (No Load)*	0.19A
Current (Stall)*	4.9A
Torque (Stall)*	27.8 oz-in (2 kgf-cm)
Gear Ratio	16:1
Gear Material	Metal
Gearbox Style	Planetary
Motor Type	DC
Output Shaft Diameter	4mm (0.1575")
Output Shaft Style	D-shaft
Output Shaft Support	Dual Ball Bearing
Electrical Connection	Male Spade Terminal
Operating Temperature	-10 ~ +60°C
Mounting Screw Size	M2 x 0.4mm
Product Weight	82g (2.89oz)

- Holonomic movement

It is possible to control an omnidirectional robot perfectly if the friction between the wheels and the floor is infinite. However, in the real world the friction and thereby the acceleration of the robot is limited. First, we show how a slipping wheel can be detected by evaluating the wheel velocities. With this information, the motor forces can be reduced to prevent slippage.



Non-holonomic machines are the ones that cannot instantaneously move in any direction. A robot like ours has wheels that can be independently controlled and they can move in any direction and rotate 360º inside its own wheelbase. Each wheel can move the robot forward, but since they are located on the periphery of the robot, they can also rotate the robot' s frame

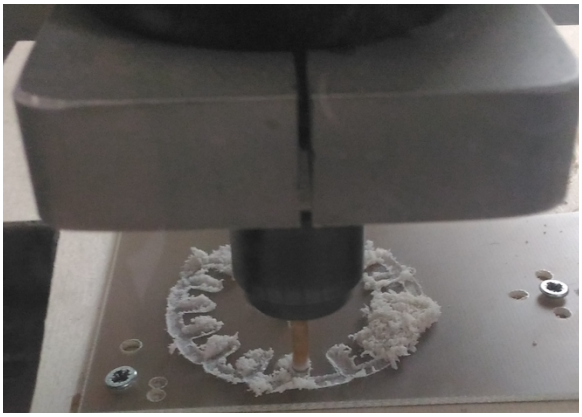
For an omni-wheel robot to translate at a particular angle, each wheel must rotate at a particular rotational velocity and direction. Since the robot is moving at angles, the software has to do trigonometric calculations to determine these wheels speeds.

Our motors are 90° apart from each other, that means that the PID controller had to be rewrited to work with the current configuration because their control is different.

- Wheels

Because of our drive system we need special wheels able to be driven with full force but also to slide laterally with great ease. Our wheels have small discs around the circumference which are perpendicular to the turning direction. so they rolls freely in two directions using the rollers mounted around its circumference We employ these wheels because. These holonomic wheels are also known as omni-wheels There are a lot of omni-weels available on the market, but we need robust ones to handle the movements. Because of the carpet on the playground, we need also ground traction and a firm grip on the ground otherwise the wheel could slip. This means that discs on each wheel must be made of rubber and their number must be higher than usual.

MAKING THE WHEELS FROM SCRATCH

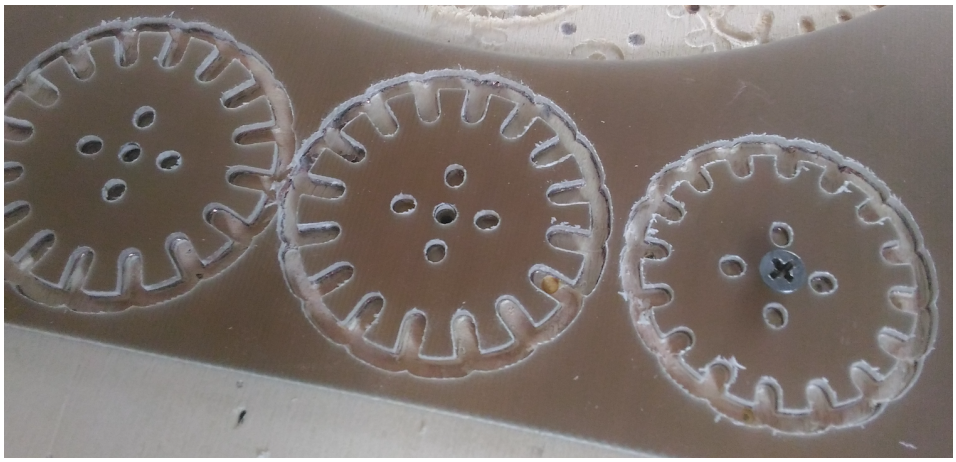


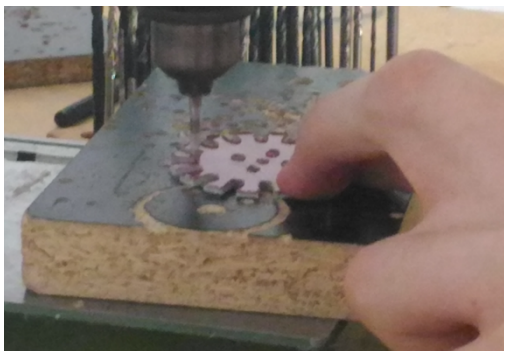
Step 1: Draw CAD for the wheels

Step 2: Export CAD for CNC

Step 3: Fix the material to be cut

Step 4: Wait for the CNC to end





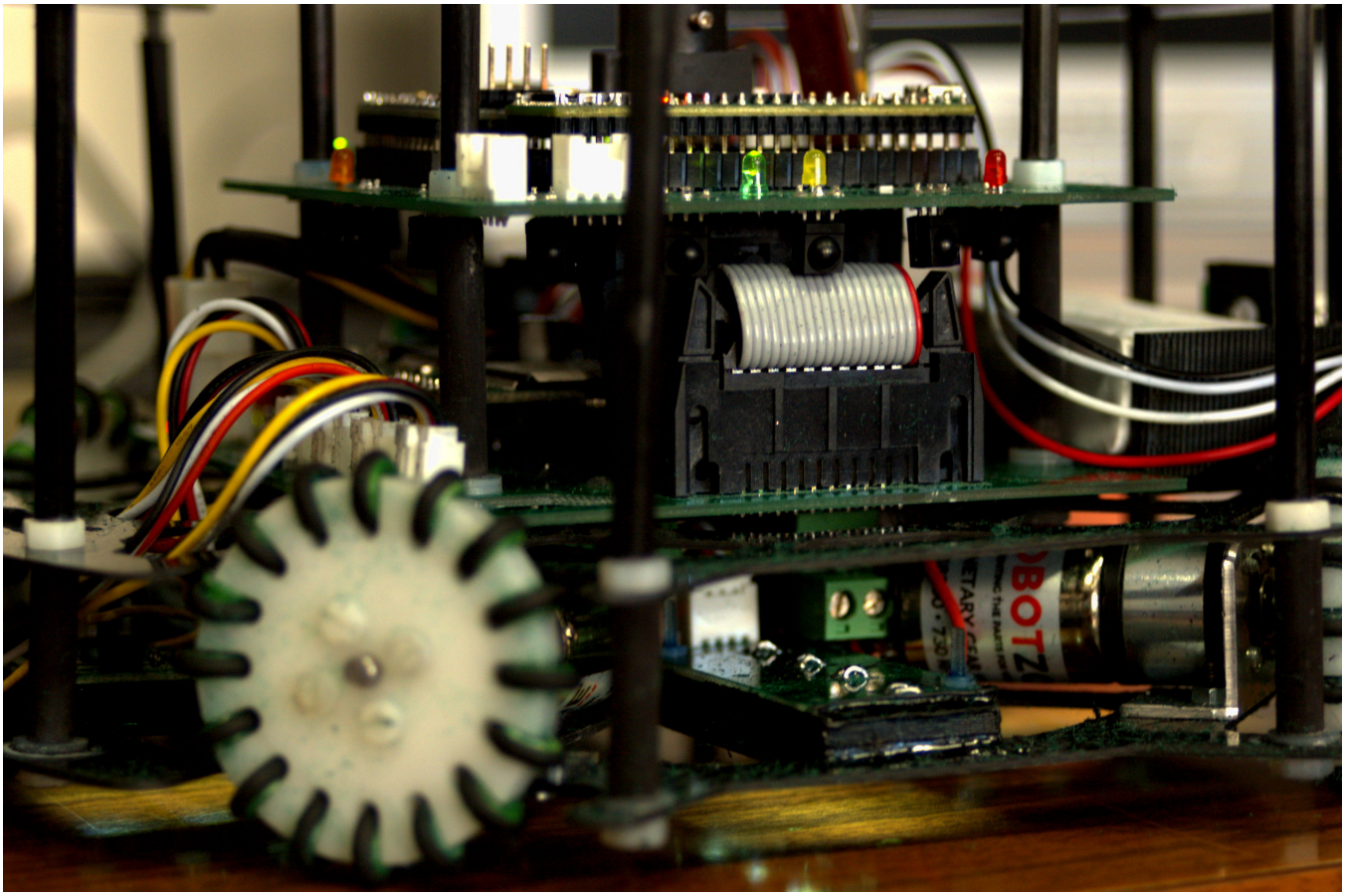
Step 5: Refine the holes

Step 6: Assemble

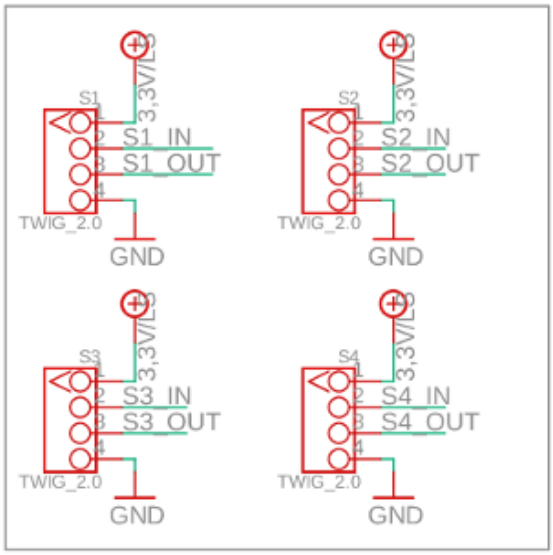
ROBO-BOARDS

Lots of innovations were brought into our new boards:

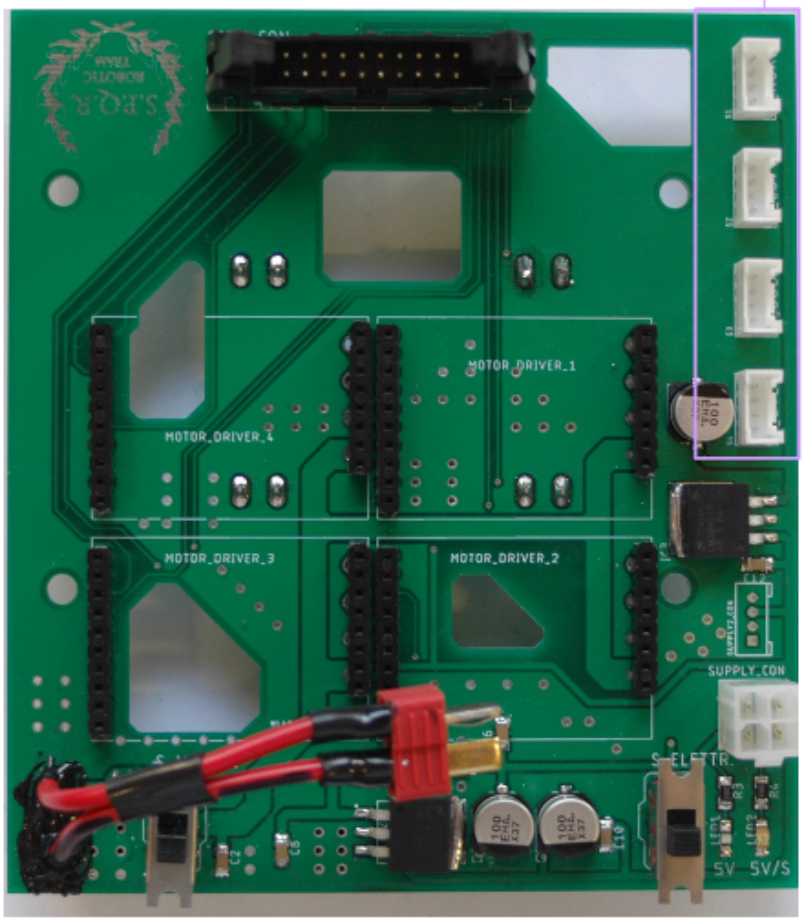
- We now use a Molex 4 pins connector for the power supply transfer because it's more secure and it can be clipped in place
- We replaced the US connectors with Grove 4 pins connectors to save space and still have functional and safer connectors
- We managed to use every single part of our PCBs, so we now mount components on the bottom as well
- Our PCBs are now squared and smaller than before: 10x10 cm and double layered to have more space to fill
- We now use four motors instead of three
- Power supply management was totally changed, as we said before

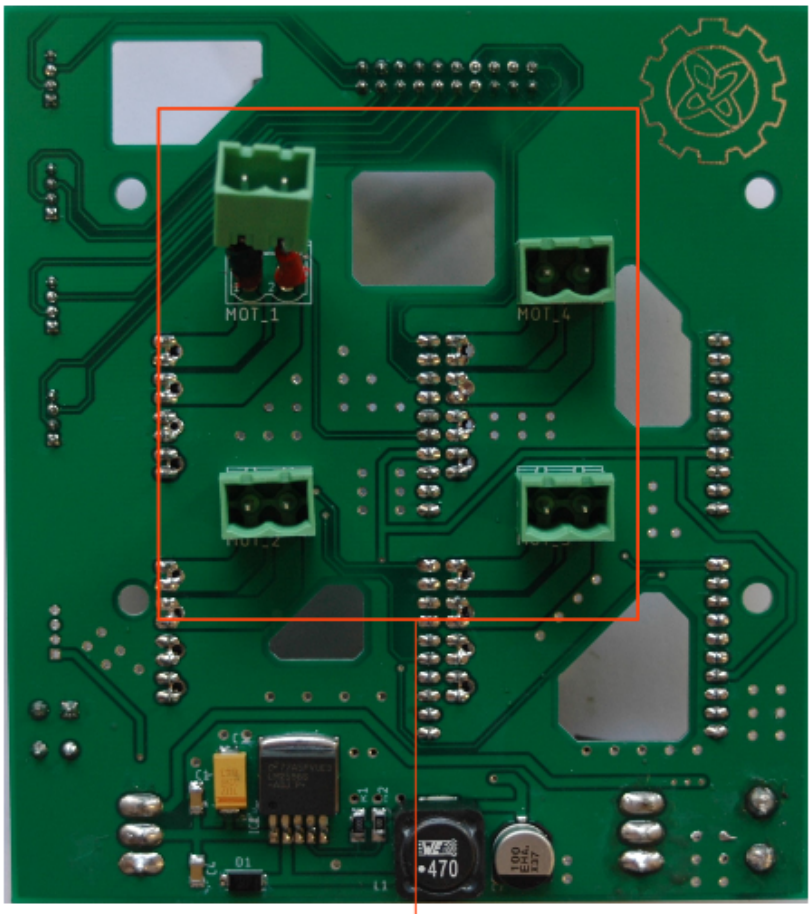


MOTOR BOARD PCB



LS_OVEST, LS_SUD, LS_EST, LS_NORD
Grove 4 pins connector used to connect the line sensors to the motor board

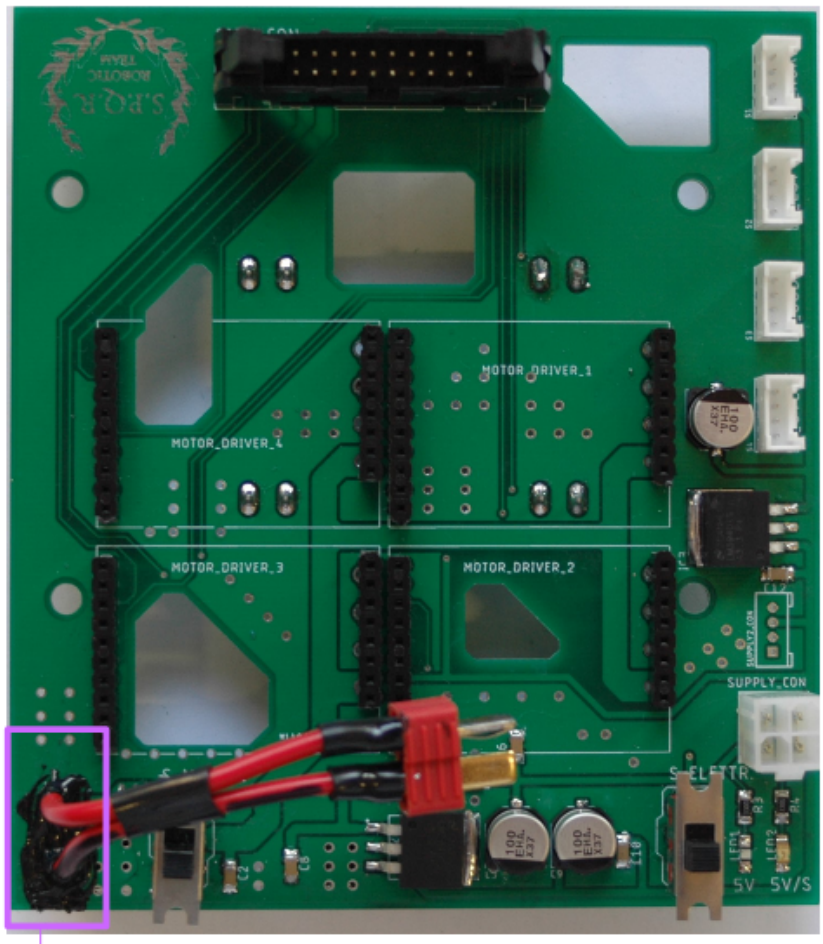




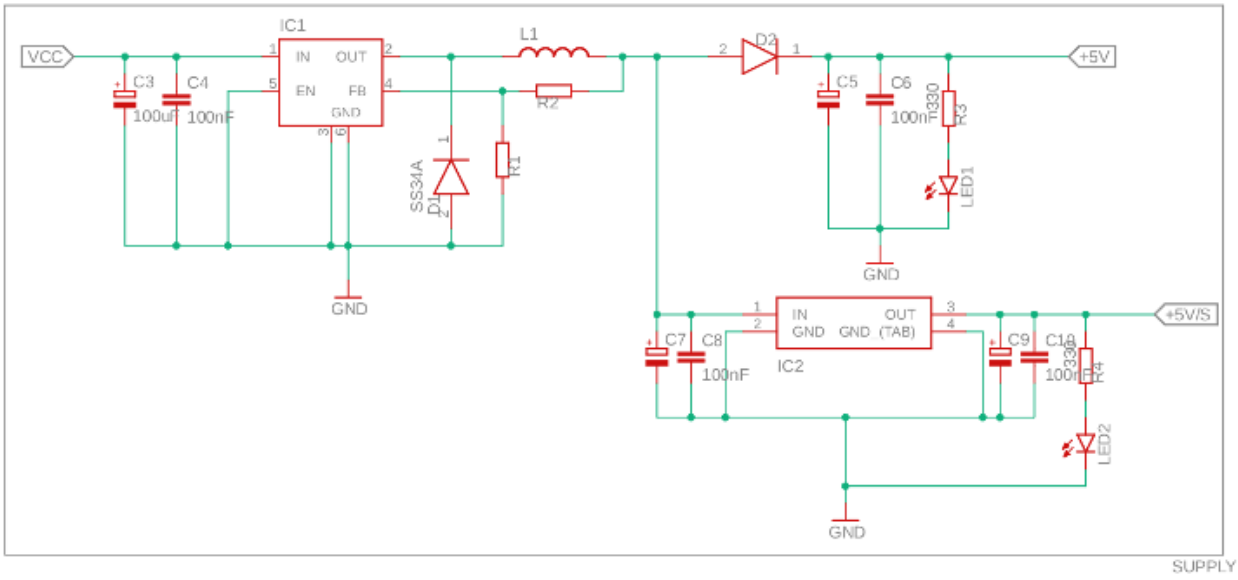
MOT_1, MOT_2, MOT_3, MOT_4

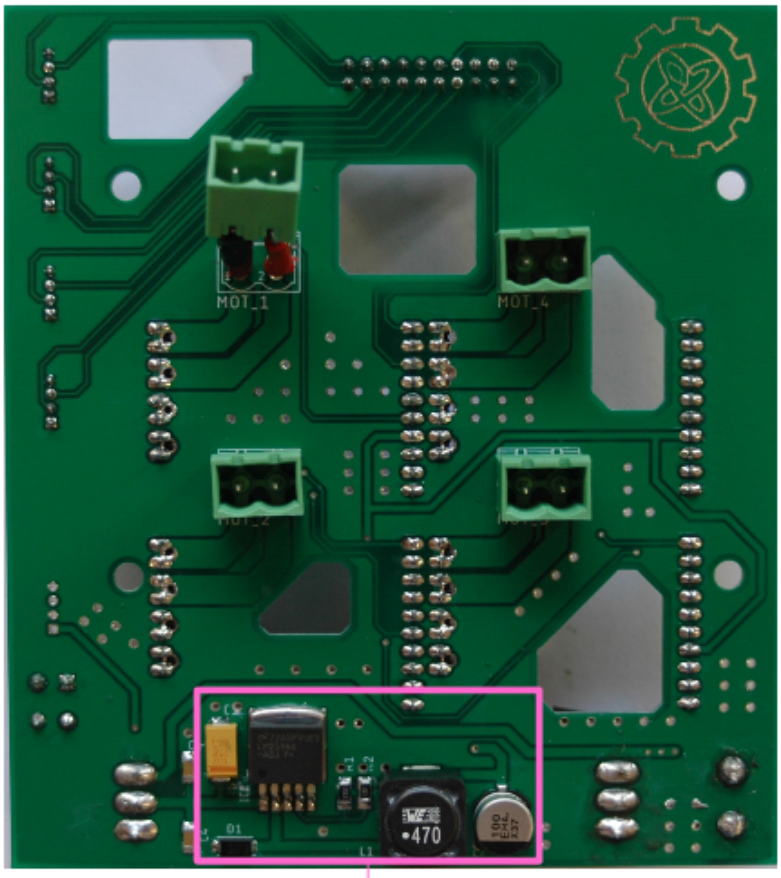
Classic Combicon MSTBA Phoenix Contact connectors are used to connect the motors to the board and the motor driver

POWER SUPPLY

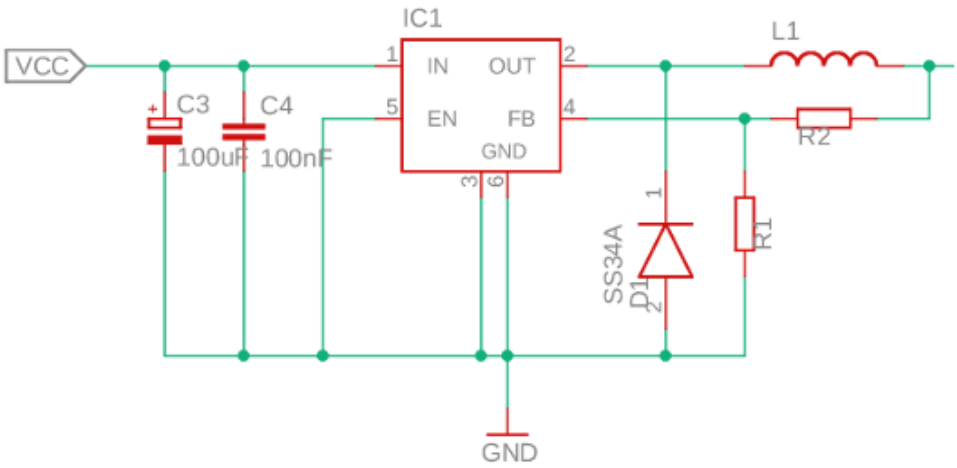


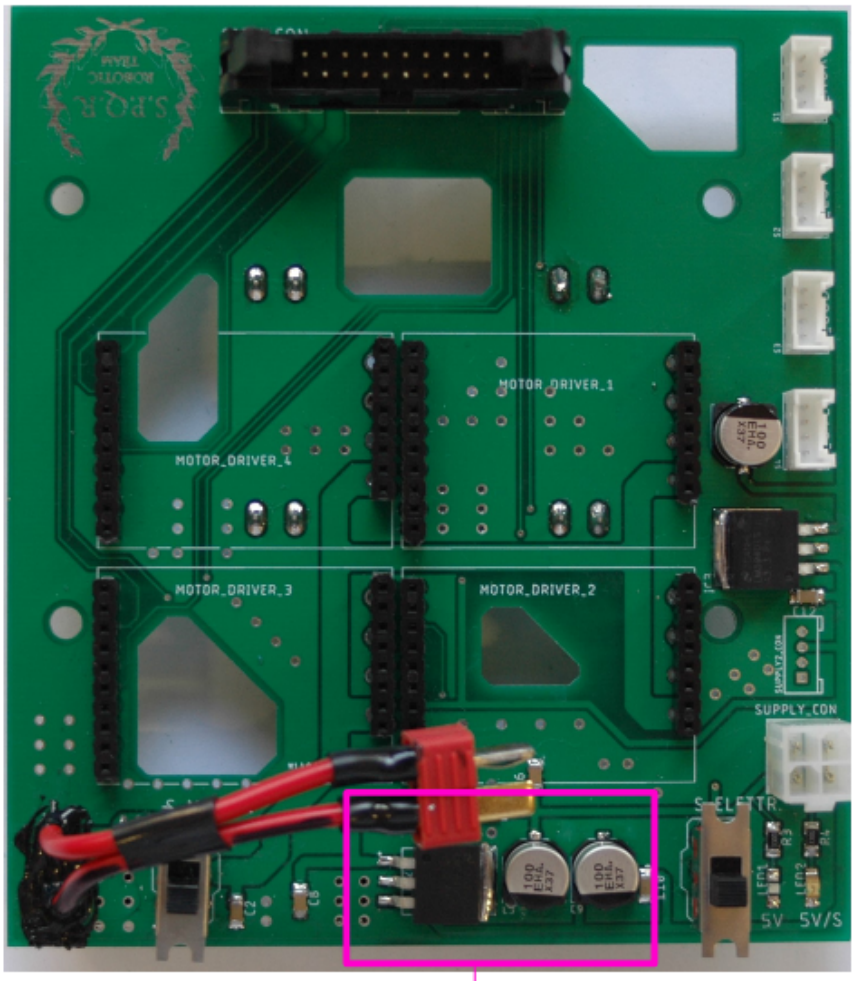
MAIN POWER SUPPLY
We use a 3 cell, 12V LiPo battery as the main power source





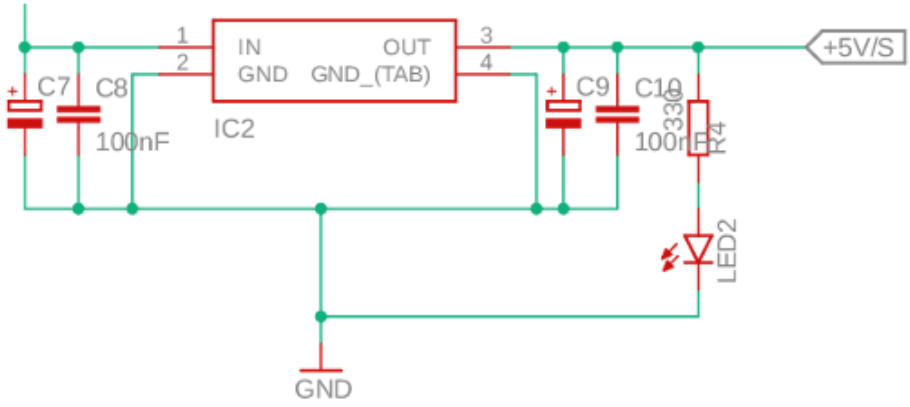
IC1
IC1 is the first voltage regulator buck converter, bringing the total voltage from 12 V, needed for the motors to work, to 5,7 V. It's made with a LM2596S-ADJ (D2PAK package), a 47 µH inductor, a 100 µF tantalium capacitor (7343-43 package) and a E SMD 100 µF capacitor

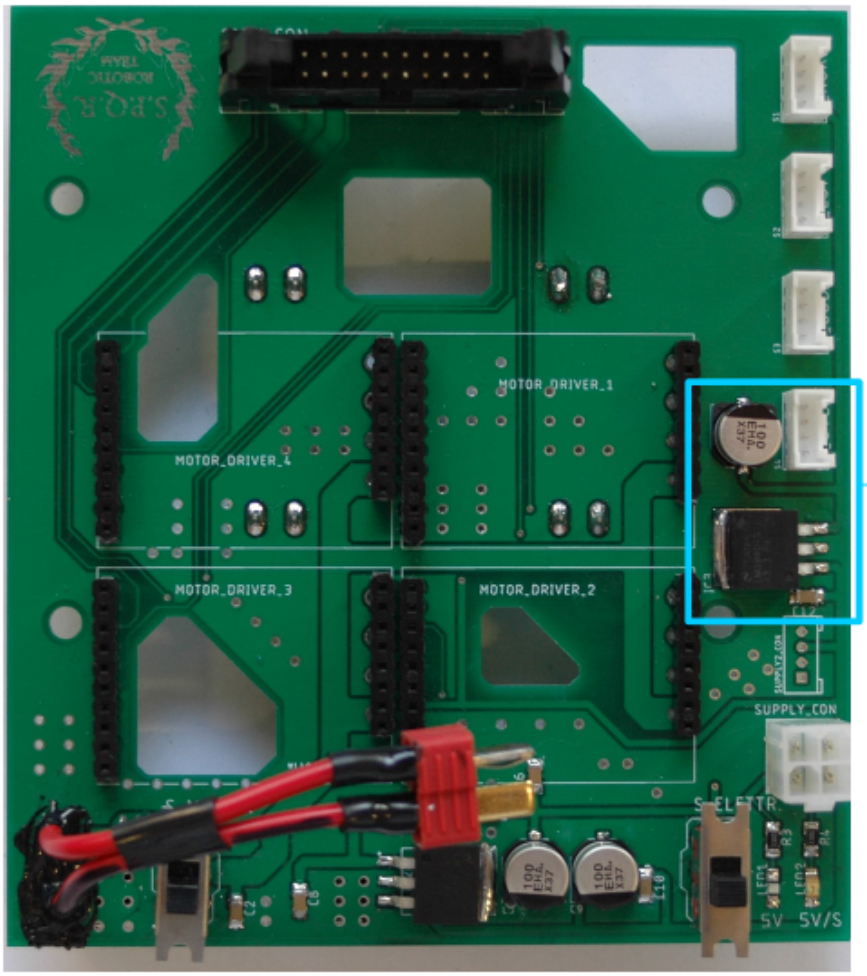




IC2

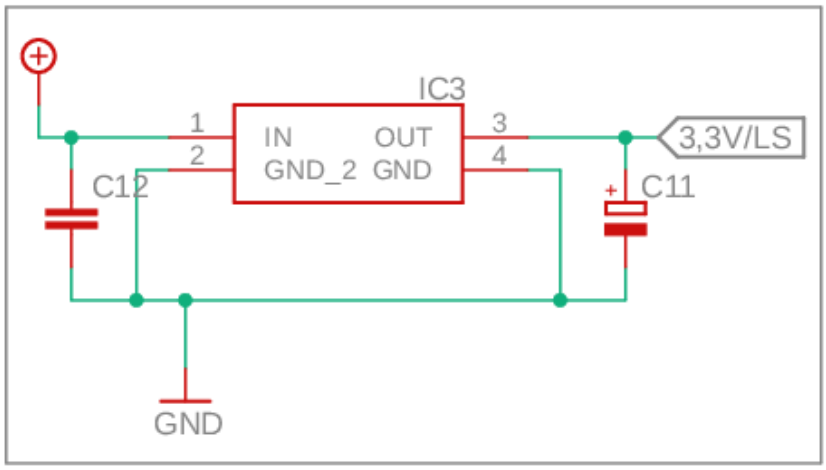
IC2 is the first voltage regulator buck converter, bringing the total voltage from 5,7 V to 5 V. It's made with aLM2940CS-5.0 (D2PAK package), a 47 µH inductor, a 100 µF tantalium capacitor (7343-43 package) and two E SMD 100 µF capacitor.





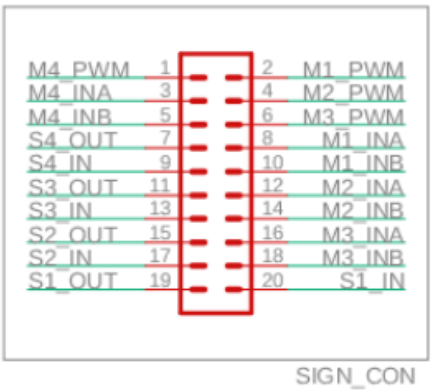
IC3

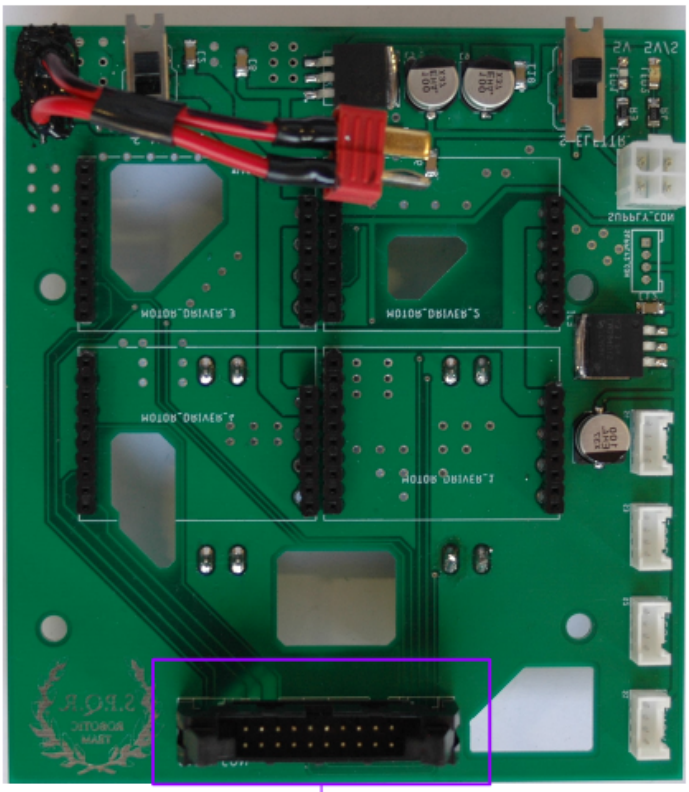
IC3 is the lines voltage regulator buck converter, bringing the total voltage from 5V to 3,3V. It's made with a LM3940IS-3.3 D2PAK and a 100 uF SMD E capacitor

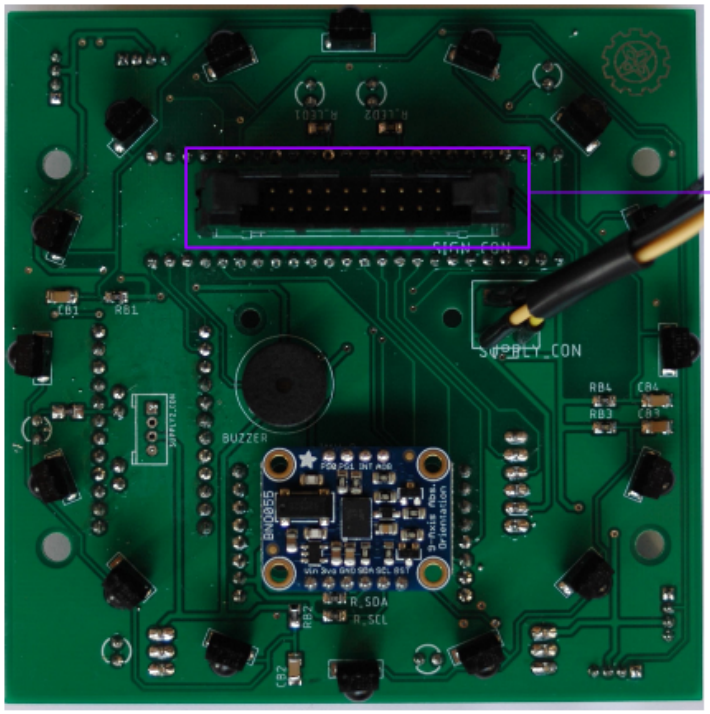


LINE_SENSORS_SUPPLY

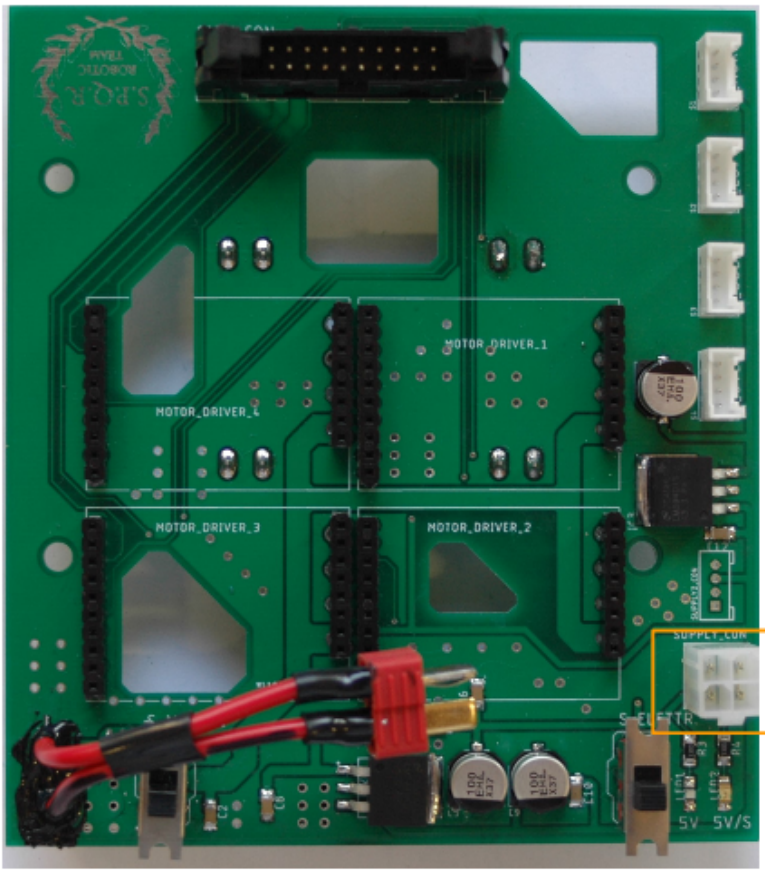
CONNECTORS BETWEEN THE BOARDS

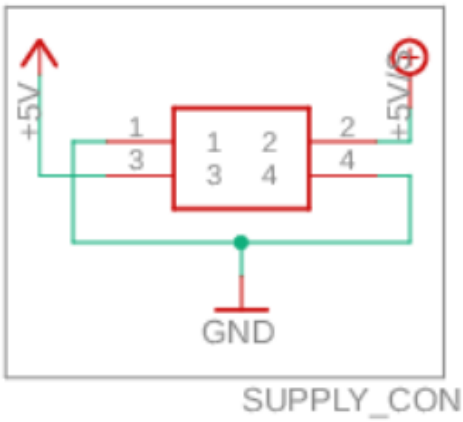


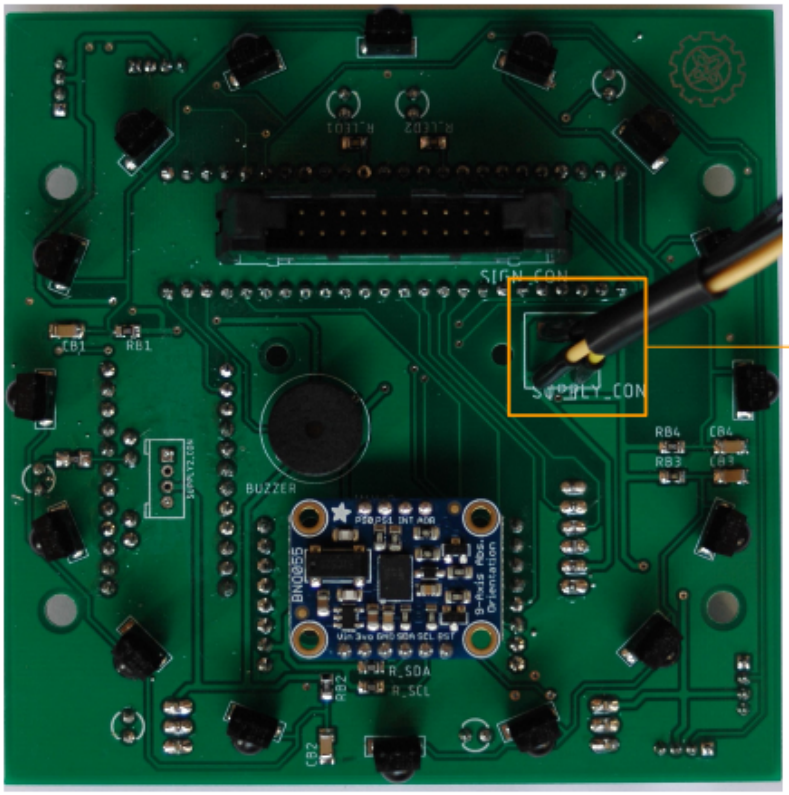




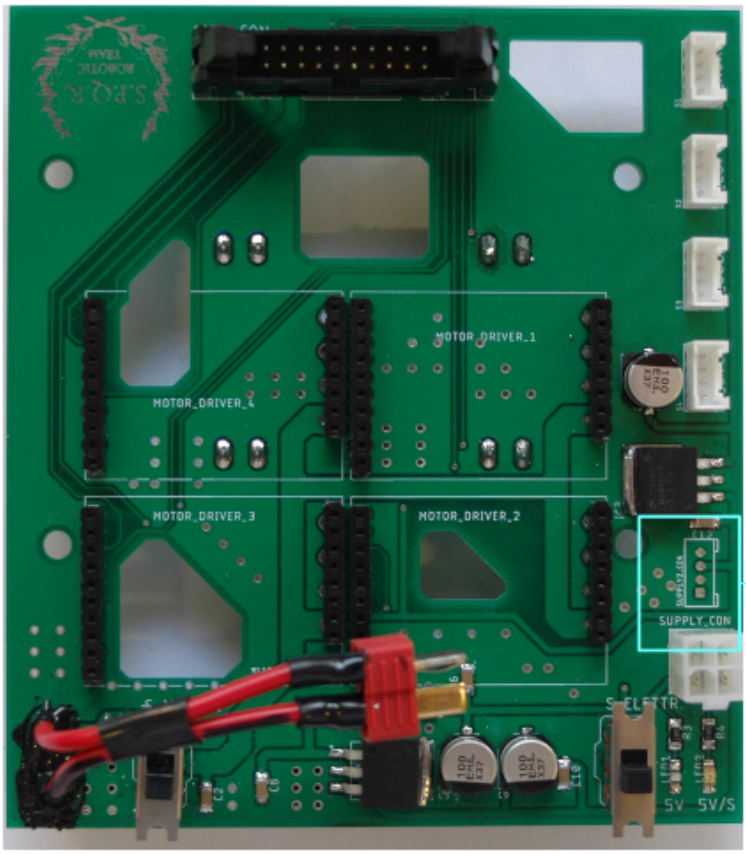
SIGN_CON
Used to transfer data from the µC board to the motor board

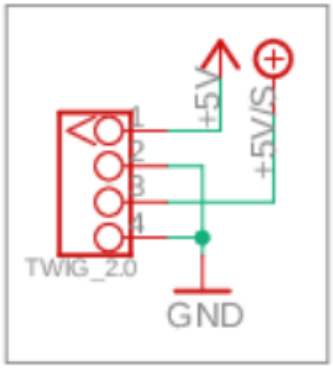


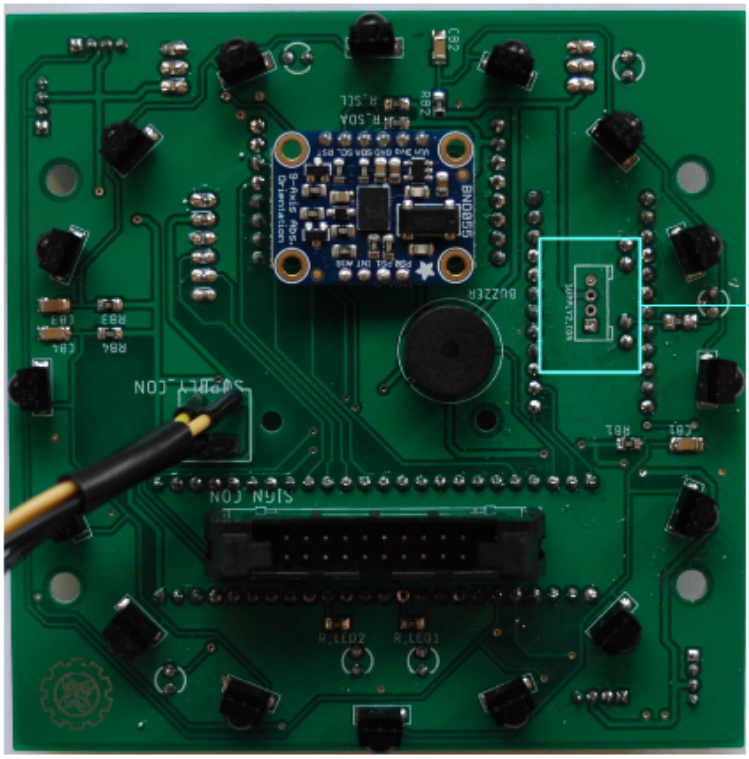




SUPPLY_CON
A 4 pins Molex connector used to transfer power from the motor board
to the µC board



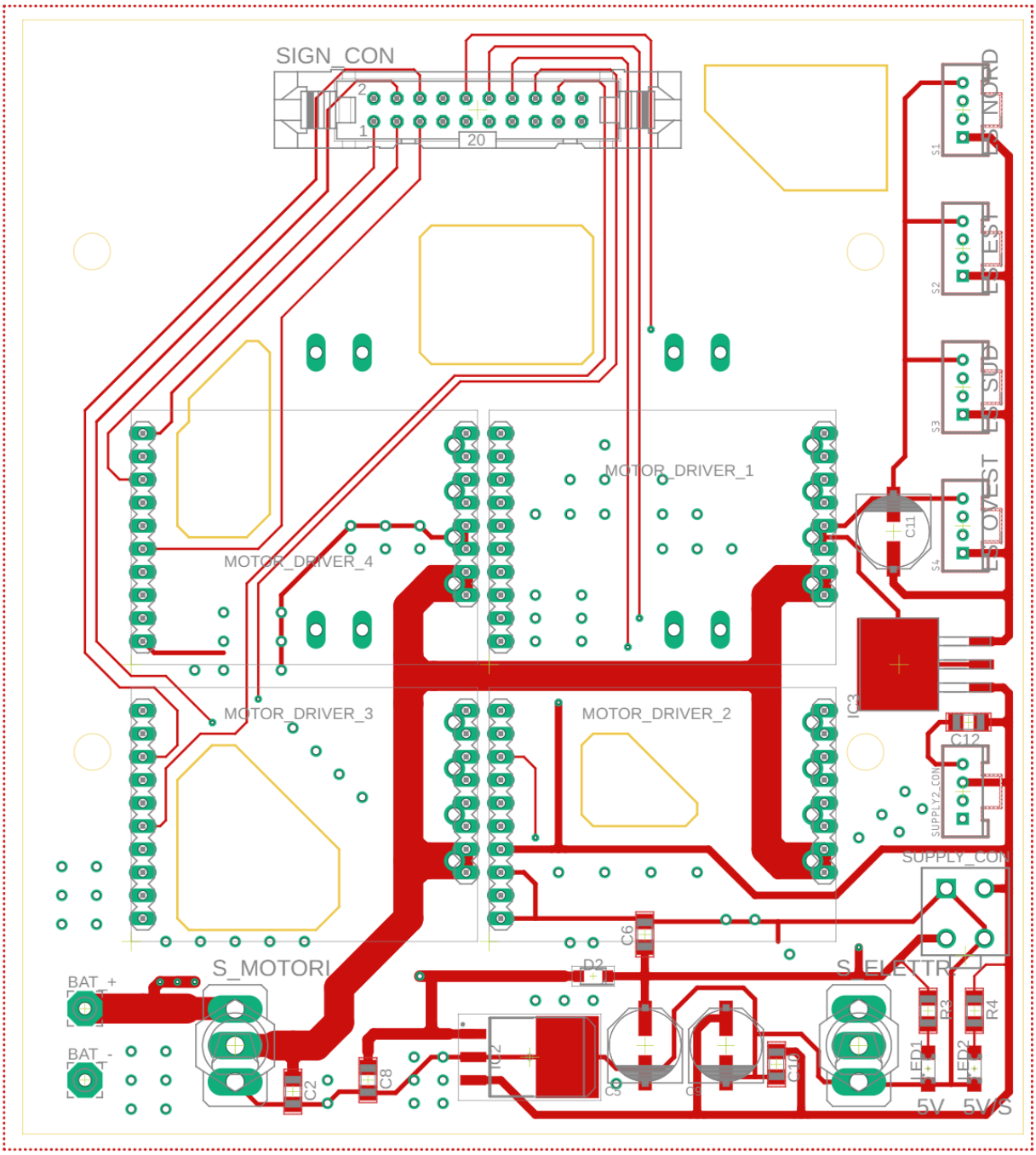




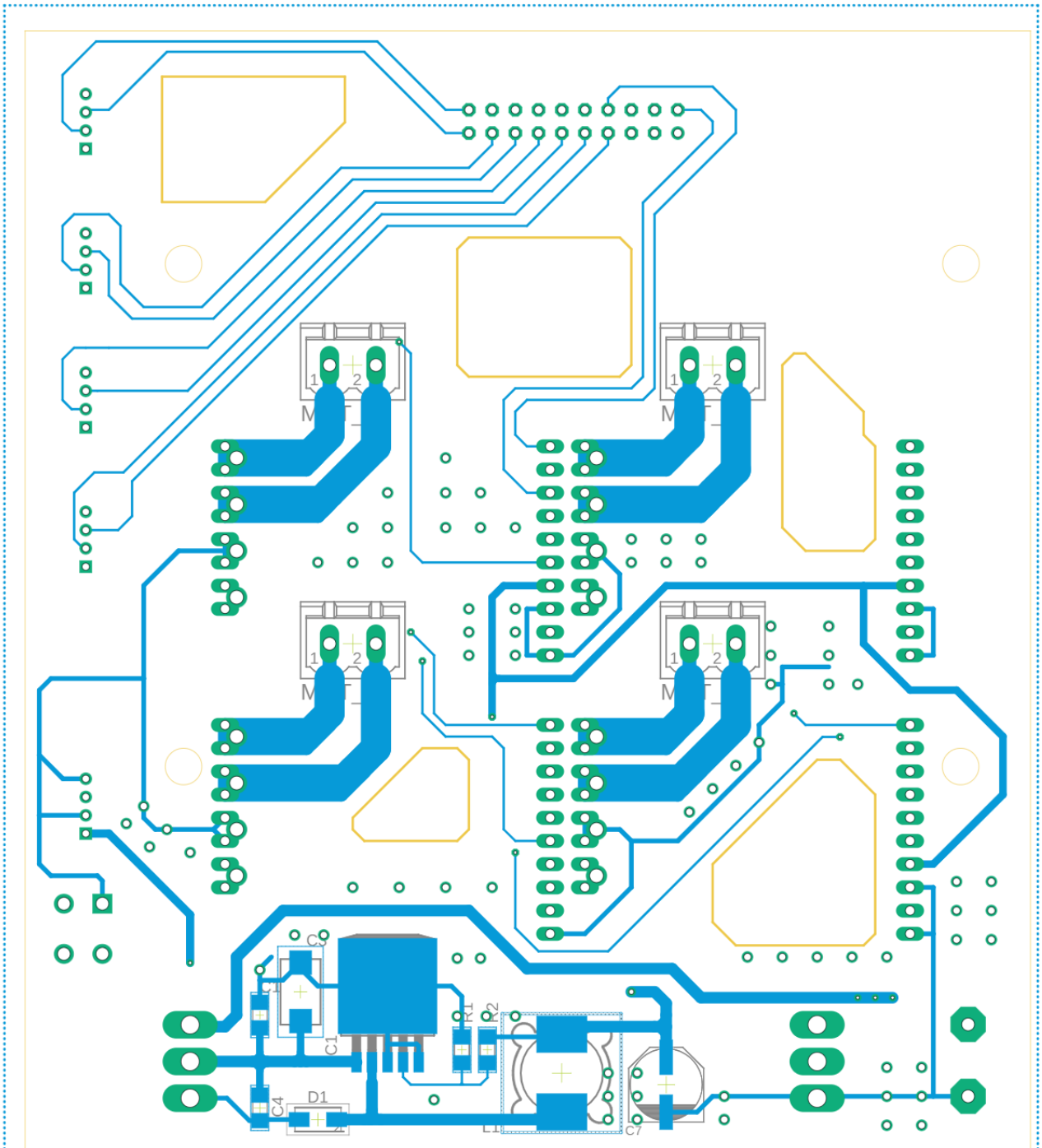
SUPPLY2_CON

Additional power supply Grove 4 pin connector, present on both boards. It's currently unused.

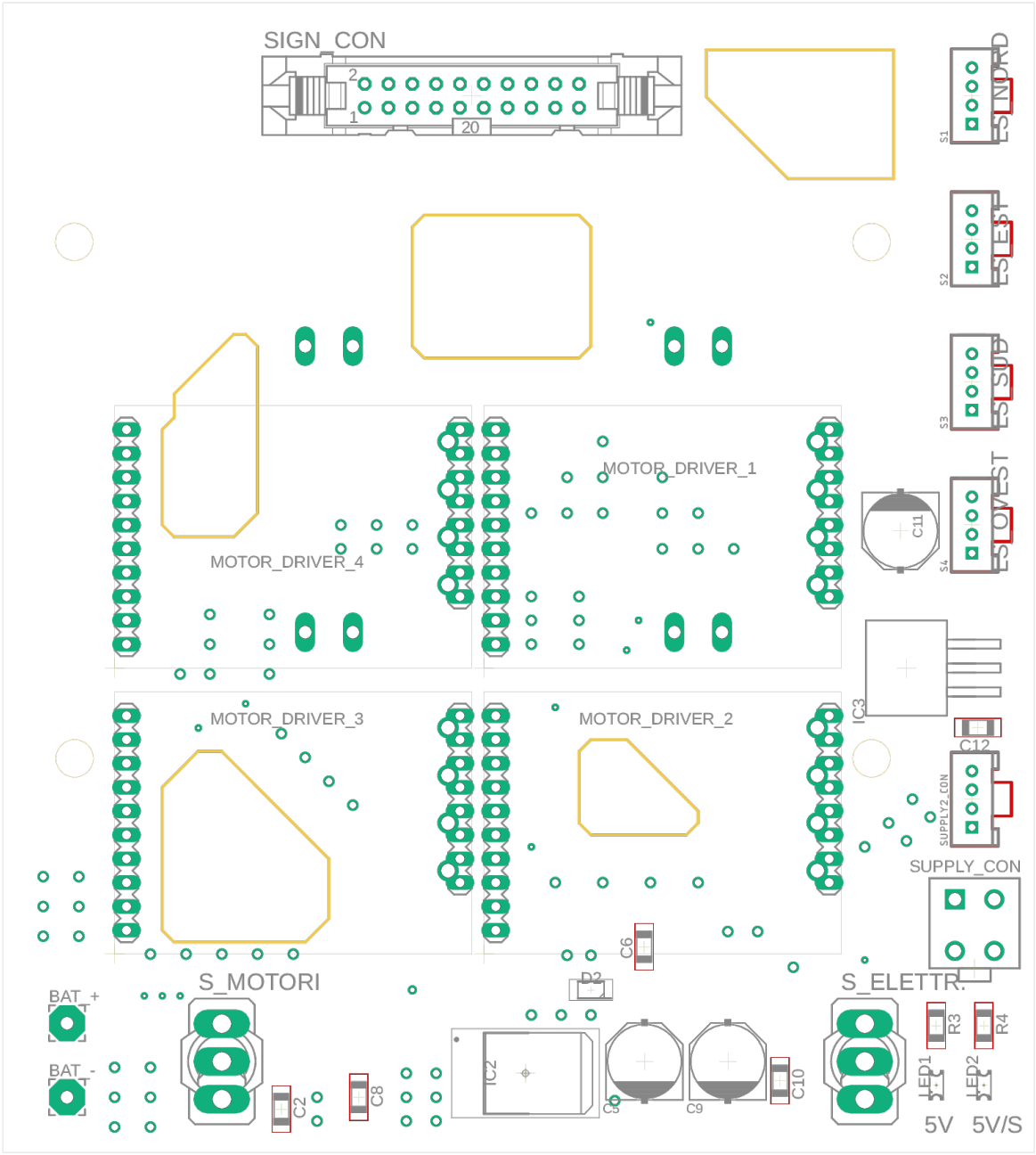
- Top


SIGN_CON
20
MOTOR_DRIVER_1
MOTOR_DRIVER_2
MOTOR_DRIVER_3
MOTOR_DRIVER_4
S_MOTORI
S_ELETTR
SUPPLY_CON
BAT_+
BAT_-
5V_5VS

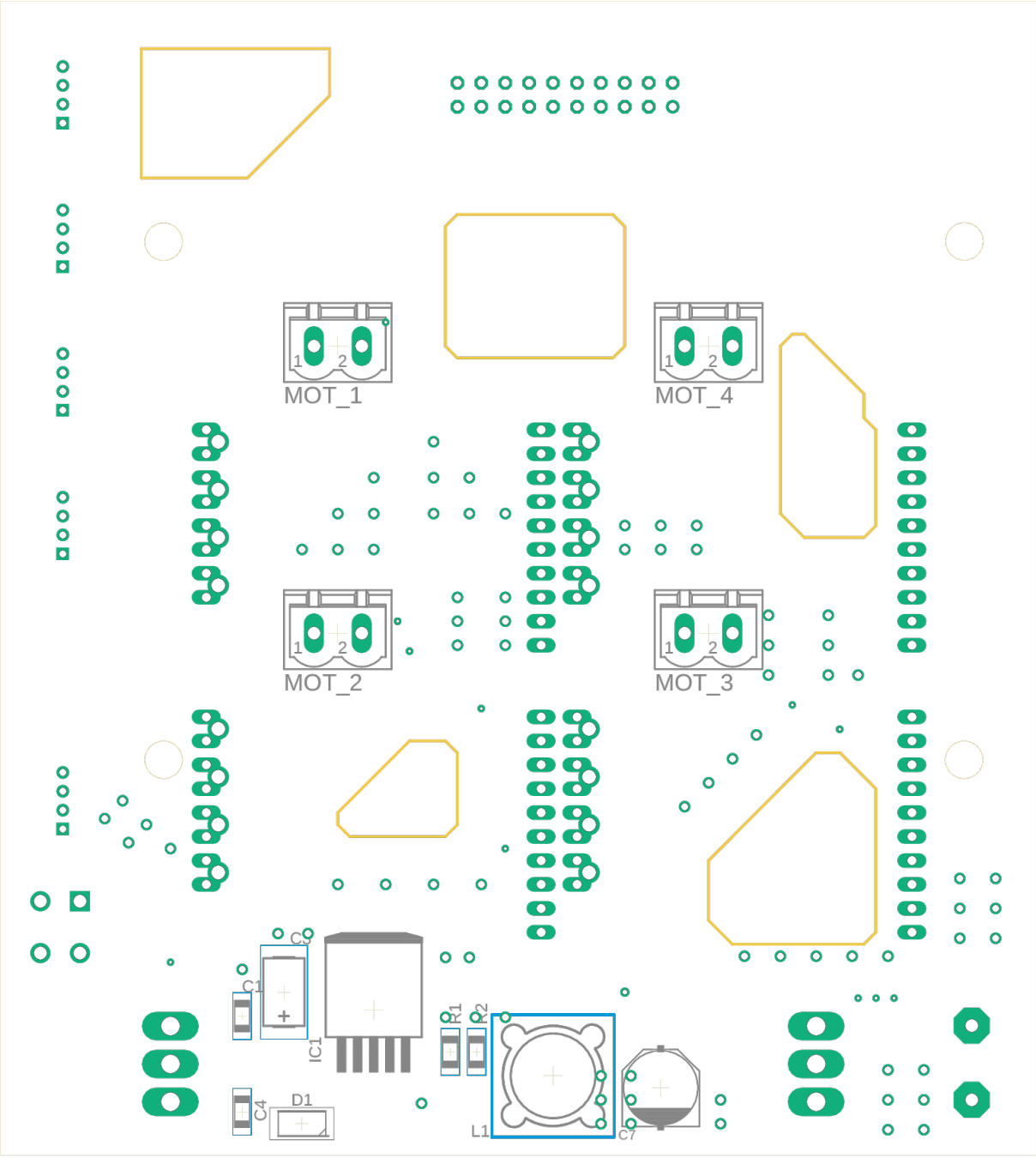
- Bottom



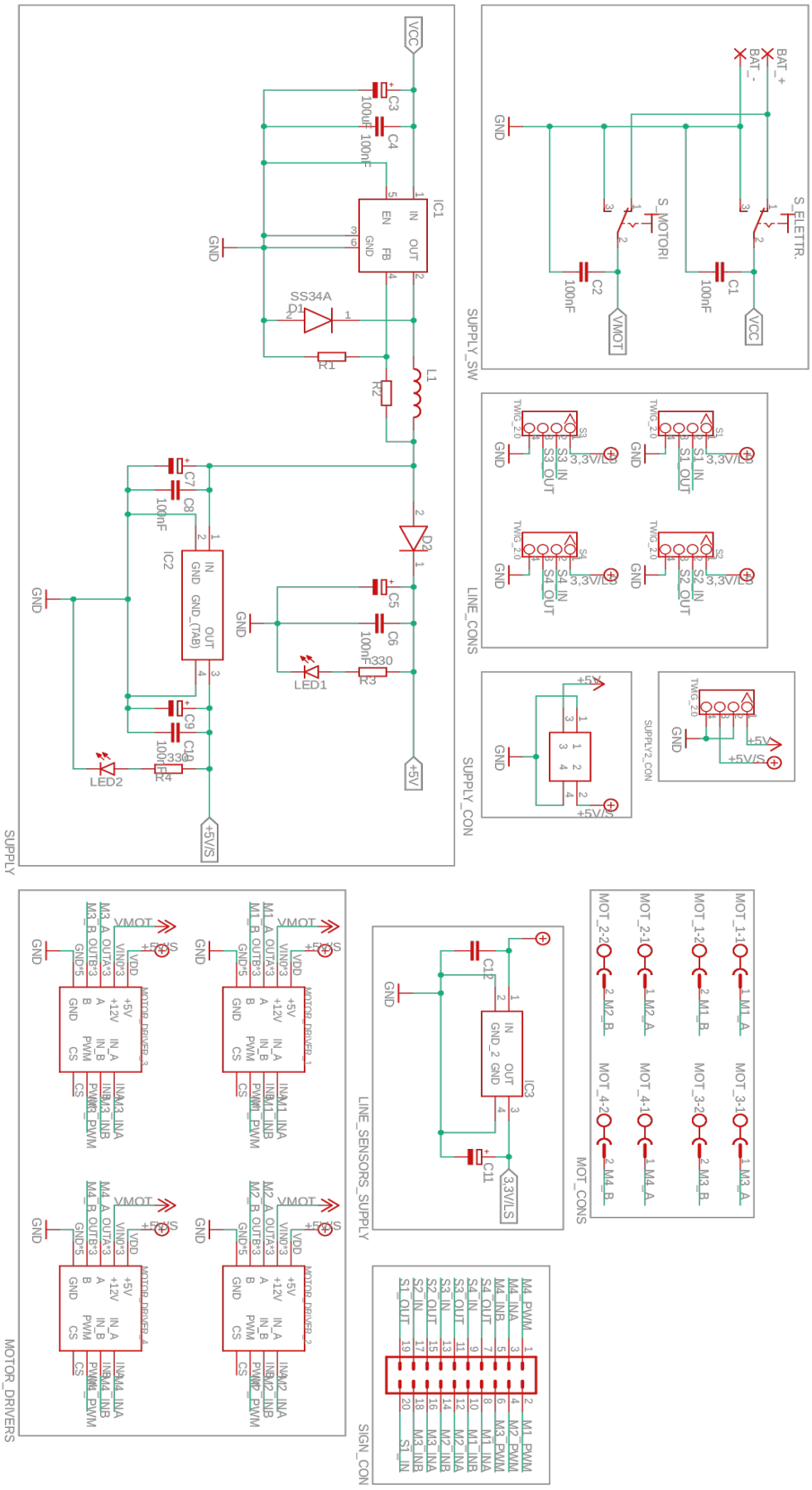
- Components layout top



- Components layout bottom

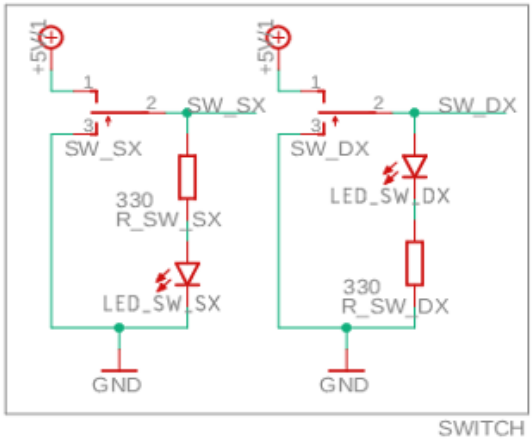


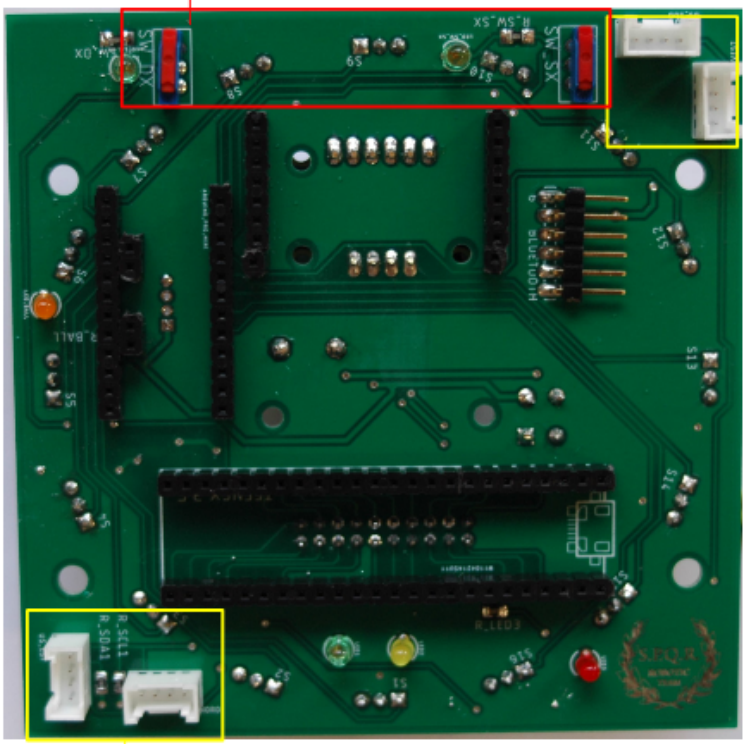
- Schematic


SUPPLY
SUPPLY_SW
LINE_CONS
SUPPLY_CON
SUPPLY2_CON
MOT_CONS
LINE_SENSORS_SUPPLY
SIGN_CON
MOTOR_DRIVERS
VCC
GND
VMOT
+5V
+5VLS
+5VS
3,3VLS
IC1
IC2
IC3
L1
D1
D2
LED1
LED2
S_ELETTR.
S_MOTORI
BAT_+
BAT_-
SS34A
MOTOR_DRIVER_1
MOTOR_DRIVER_2
MOTOR_DRIVER_3
MOTOR_DRIVER_4

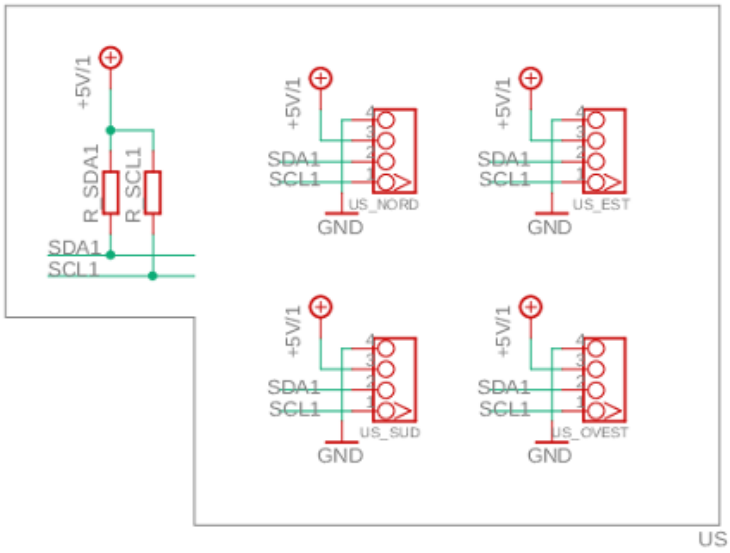
MICROCONTROLLER BOARD PCB

SW_SX, SW_DX
Status switches, used to change the role and the goalpost

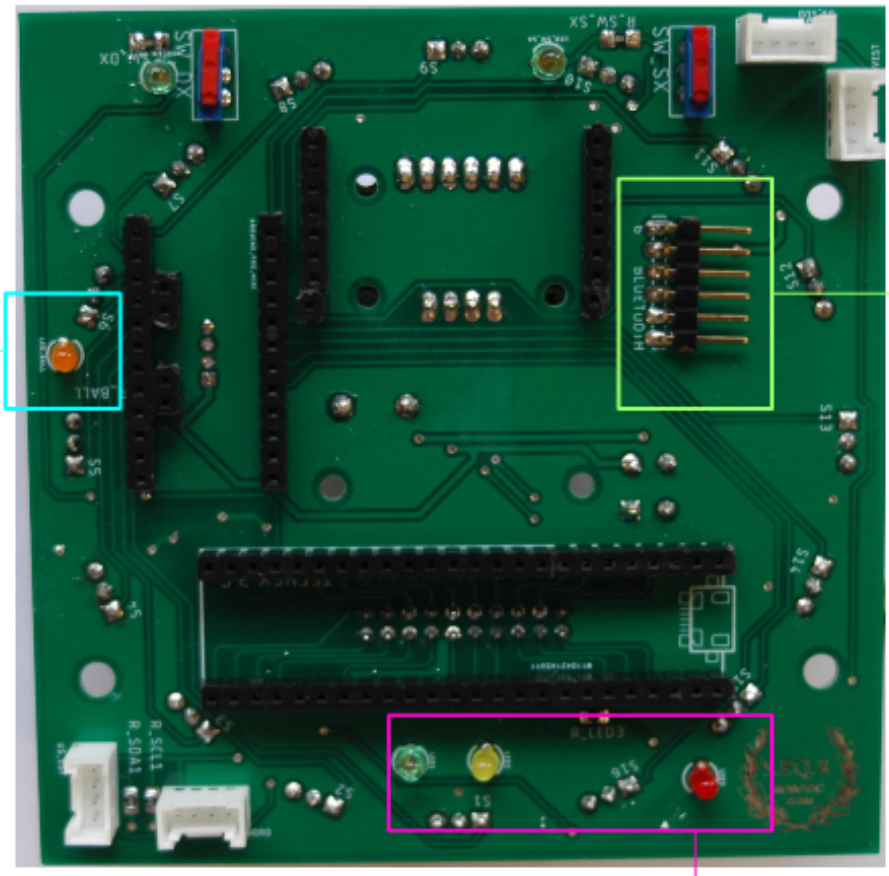




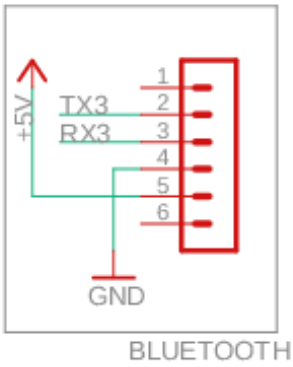
US_NORD, US_SUD, US_OVEST, US_EST
Grove 4-pin connectors to connect the us sensors to the board



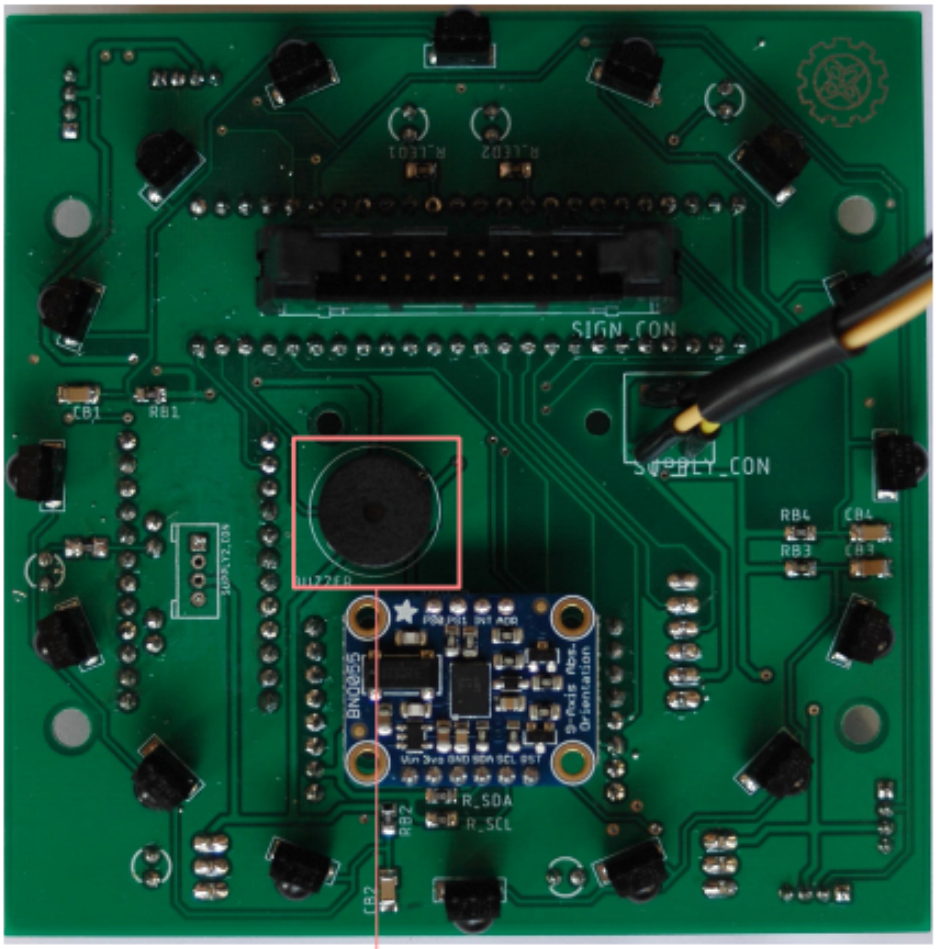
LED_BALL
Used to know when the ball is seen



BLUETOOTH
Angled connector for the BT

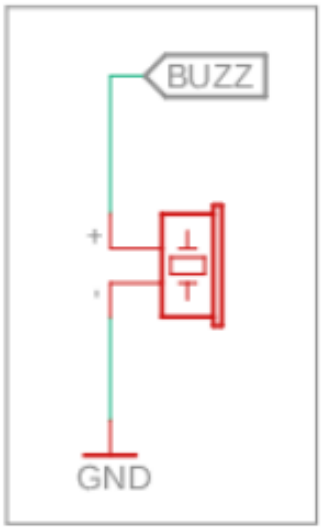


LED1, LED2, LED3
Debugging LEDs



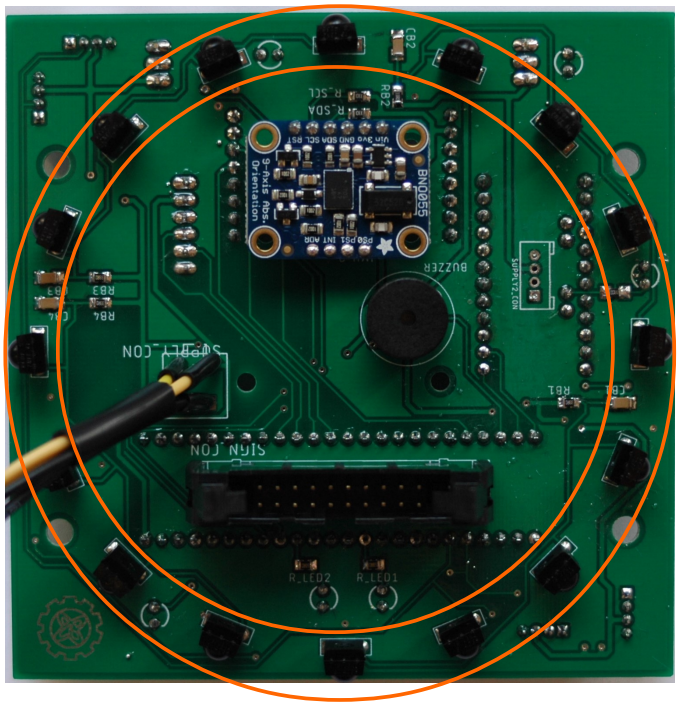
BUZZER

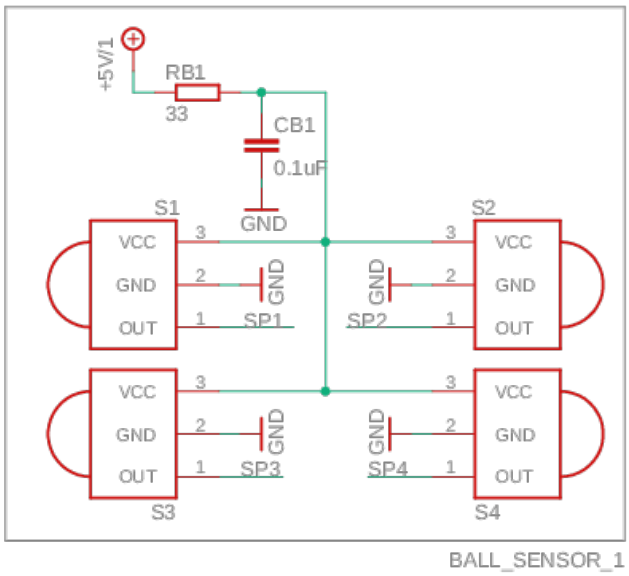
Used to make sounds in the line handling and when the robot is turned on



BUZZER

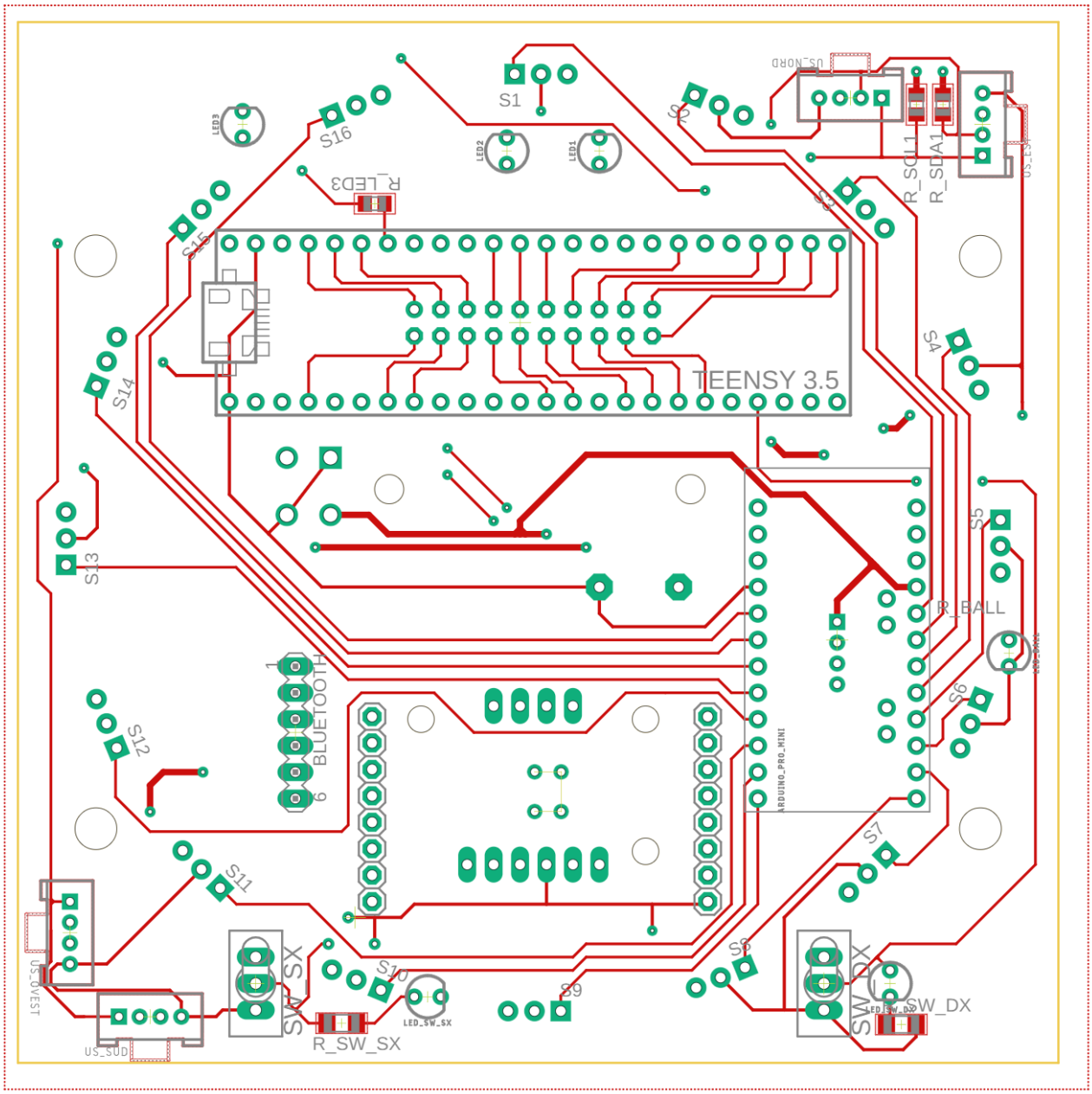
BALL SENSORS



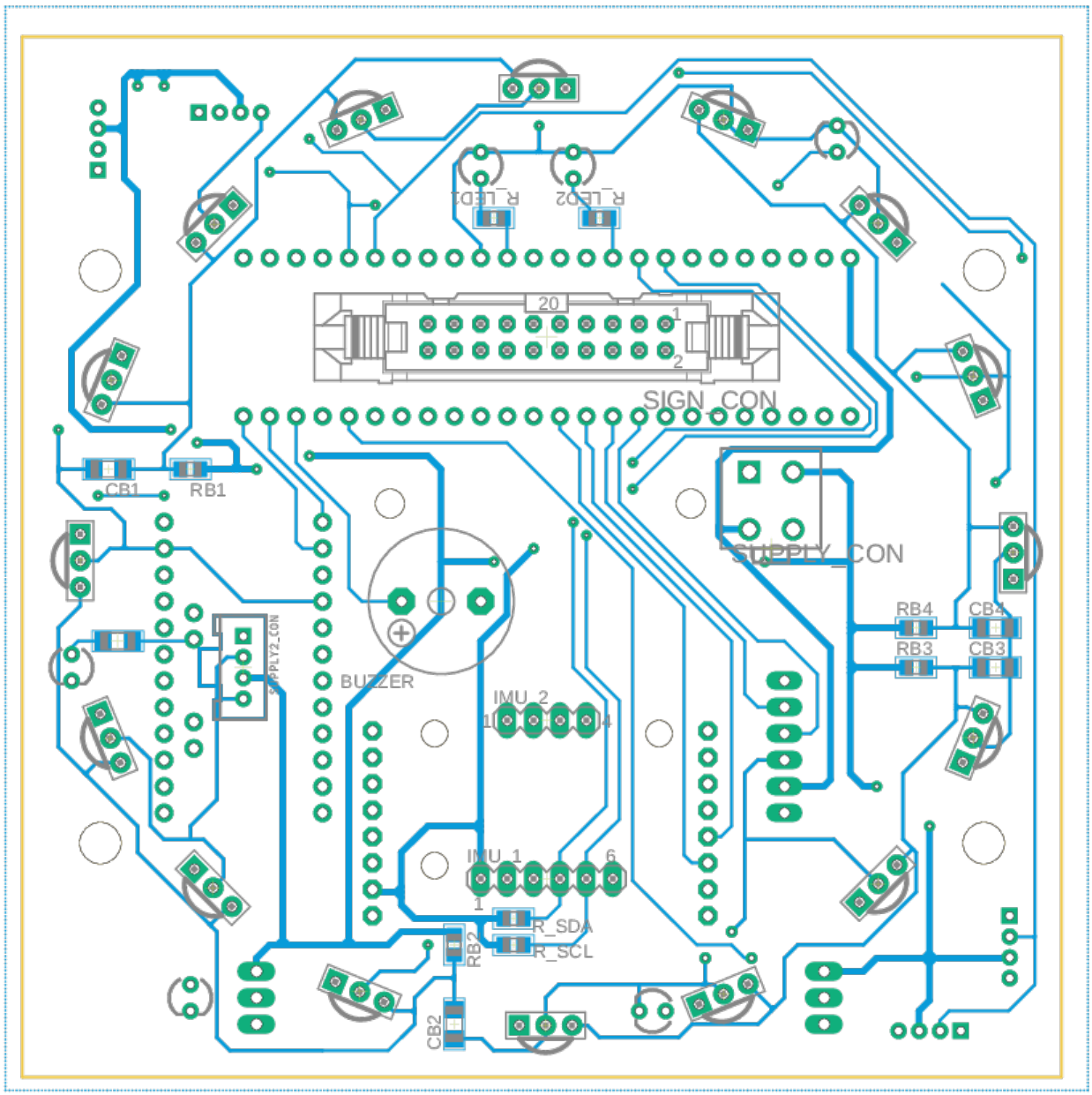


We use 16 ball sensors to fully see the ball in the field. We removed the pipes because with smaller PCBs they were floating, making the structure really unstable in the end. The received data from the sensors is now interpolated them via software. The sensors are handled by our slave microcontroller: Arduino Pro Mini, who exchanges data with the Teensy.

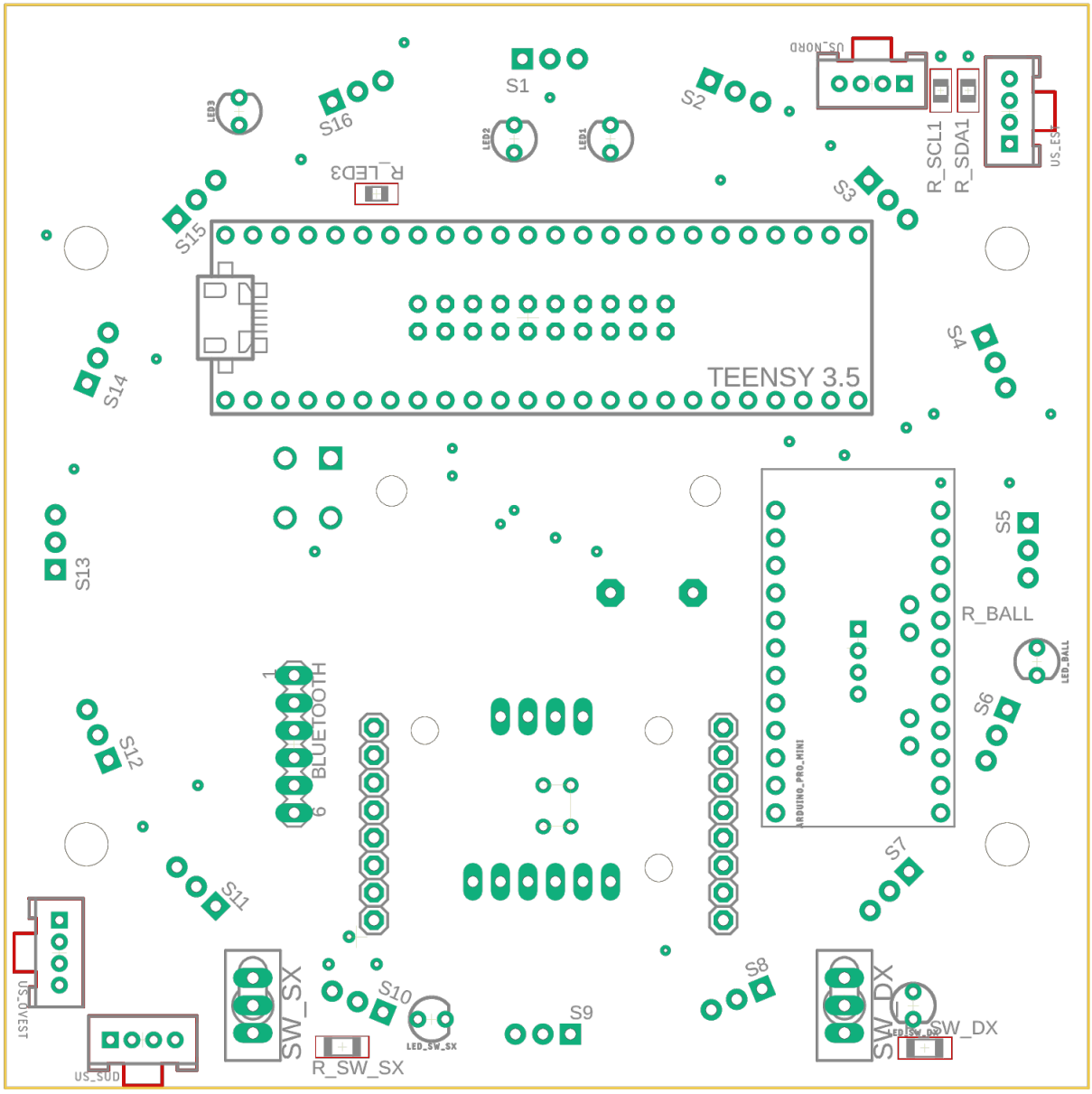
- Top


TEENSY 3.5
BLUETOOTH
R_BALL
R_SW_SX
R_SW_DX
R_LED3
R_SCL1
R_SDA1
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
LED1
LED2
LED3
LED_SW_SX
LED_SW_DX
US_NORD
US_OVEST
US_SUD
US_EST
ARDUINO_PRO_MINI

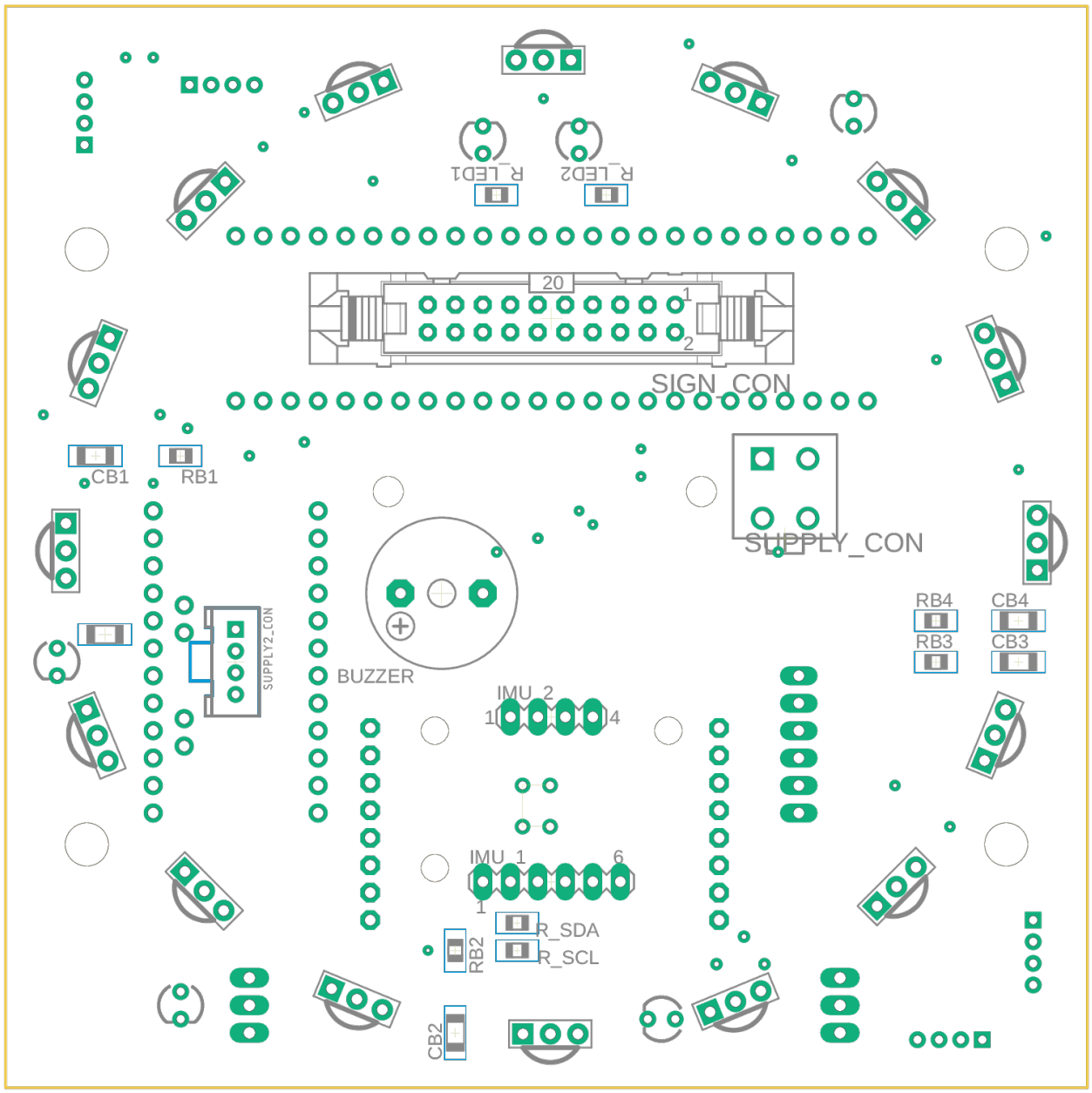
- Bottom


R_LED1
R_LED2
20
1
2
SIGN_CON
CB1
RB1
SUPPLY_CON
RB4
CB4
RB3
CB3
SUPPLY2_CON
BUZZER
IMU_2
IMU_1
6
R_SDA
R_SCL
RB2
CB2

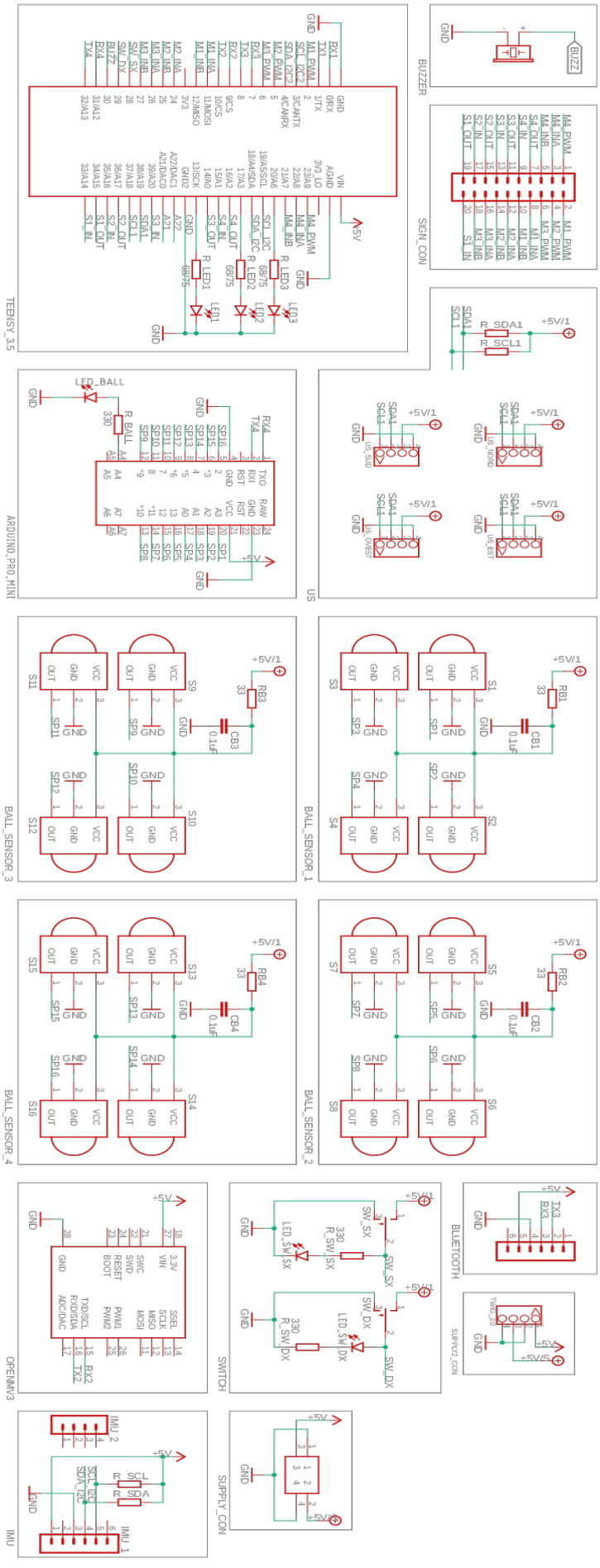
- Components layout top



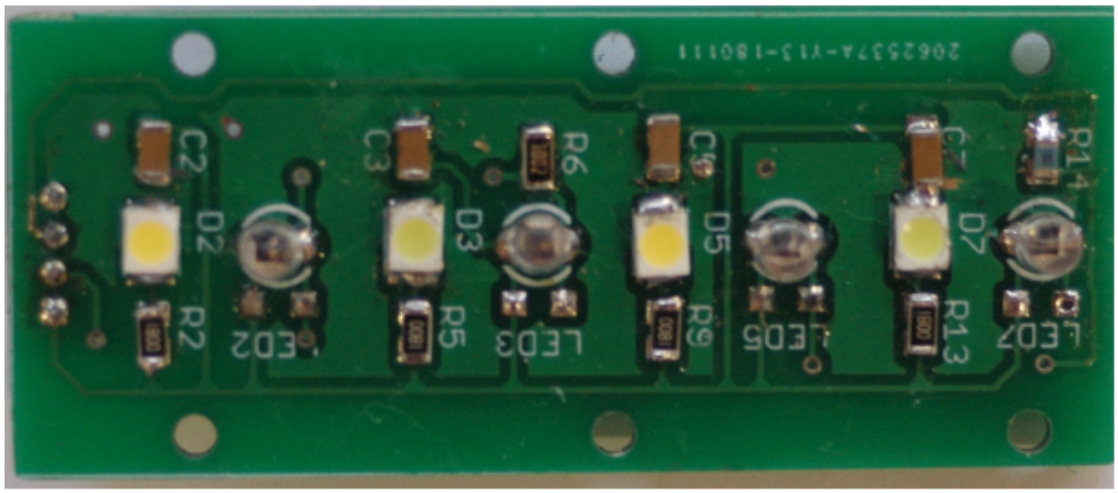
- Components layout bottom

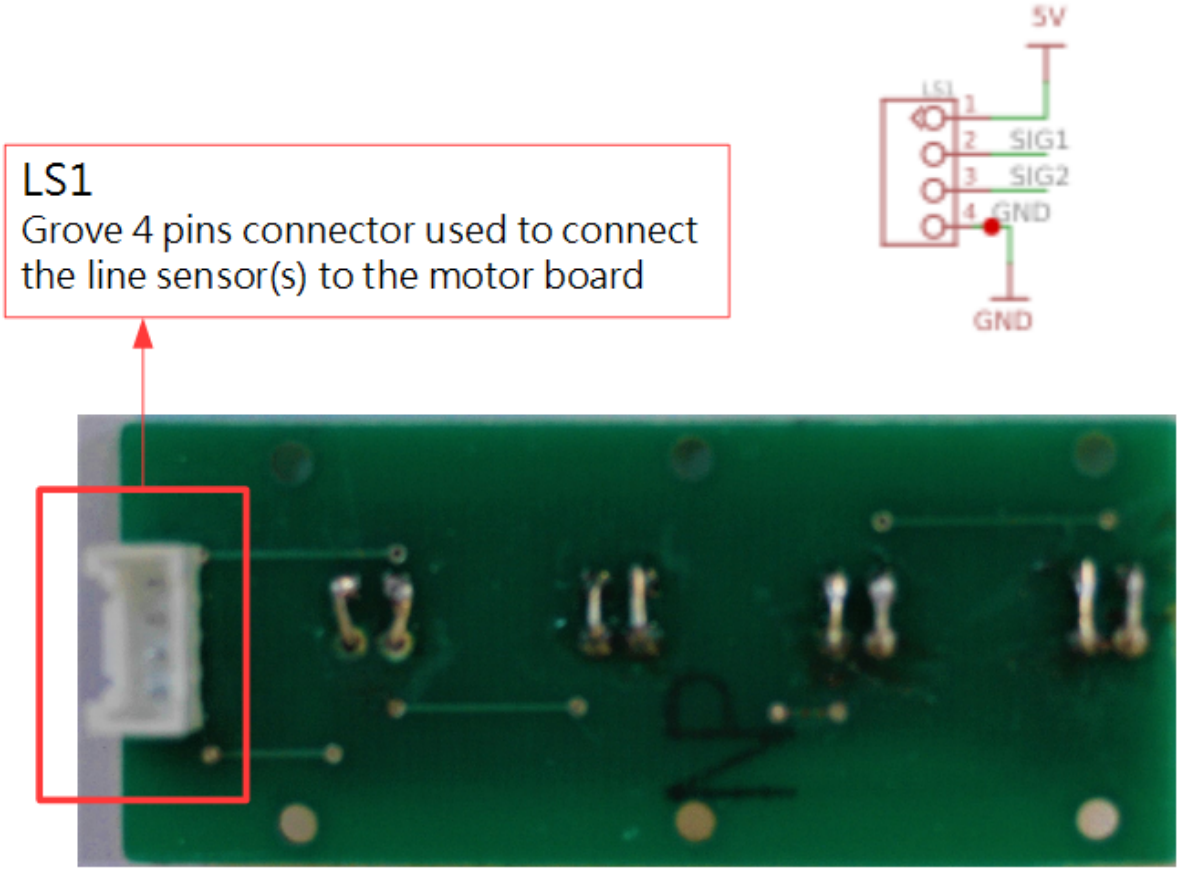


- Schematic

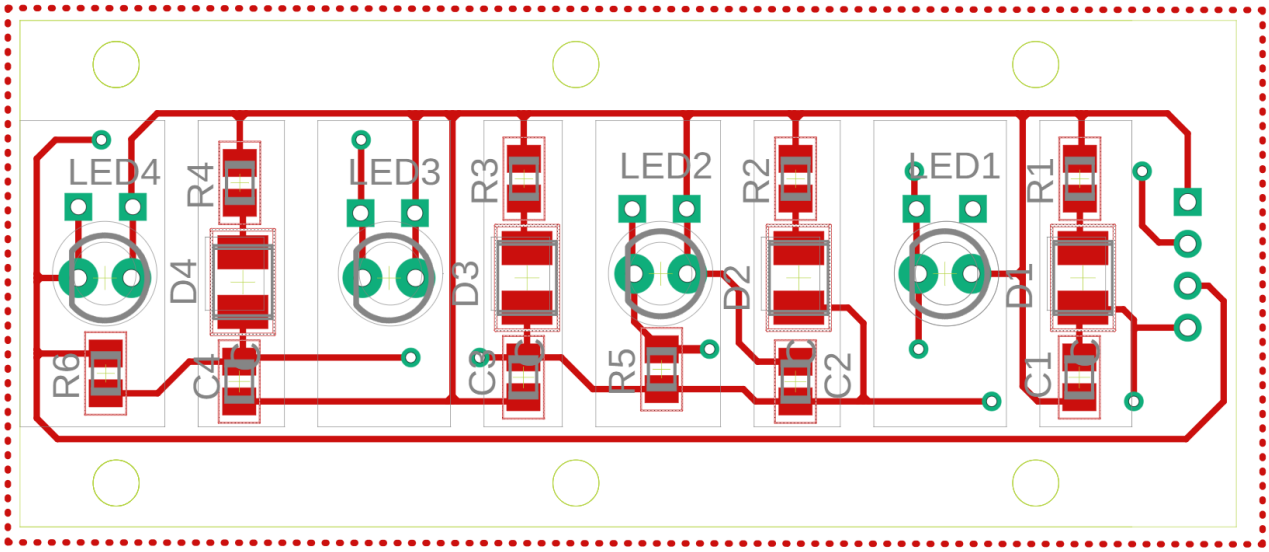

TEENSY_3.5
BUZZER
SIGN_CON
US
ARDUINO_PRO_MINI
BALL_SENSOR_3
BALL_SENSOR_1
BALL_SENSOR_4
BALL_SENSOR_2
BLUETOOTH
SUPPLY2_CON
OPENMV3
SWITCH
SUPPLY_CON
IMU

LINE SENSORS PCB

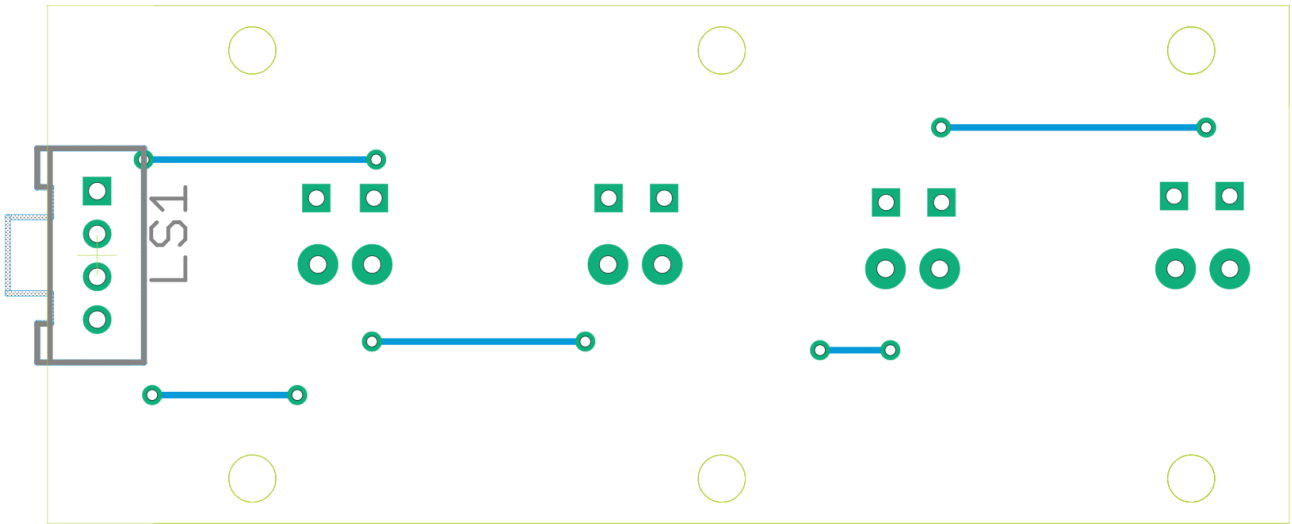




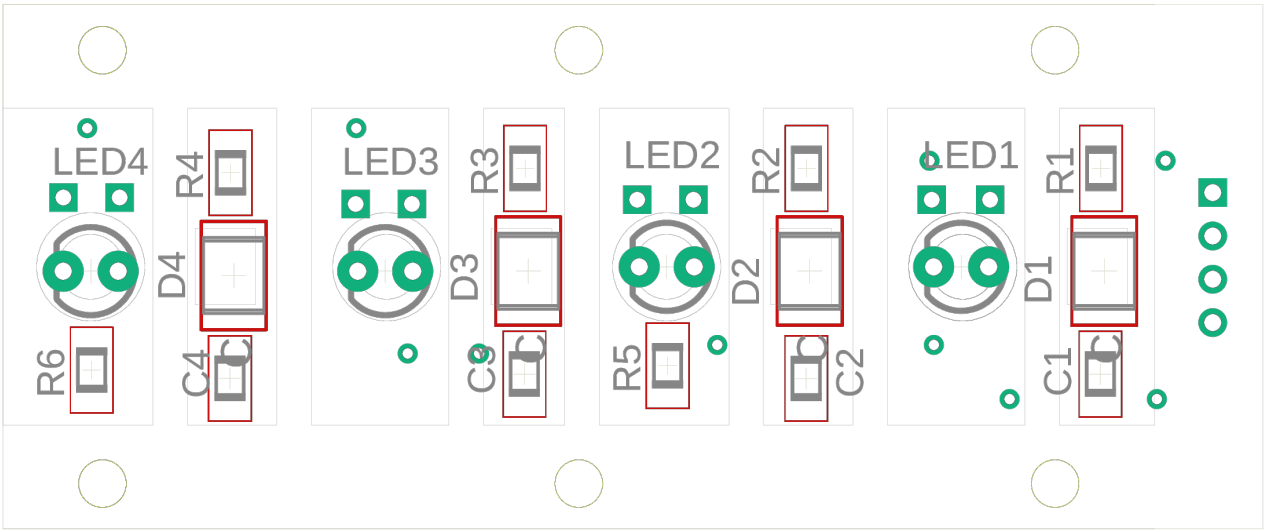
- Top



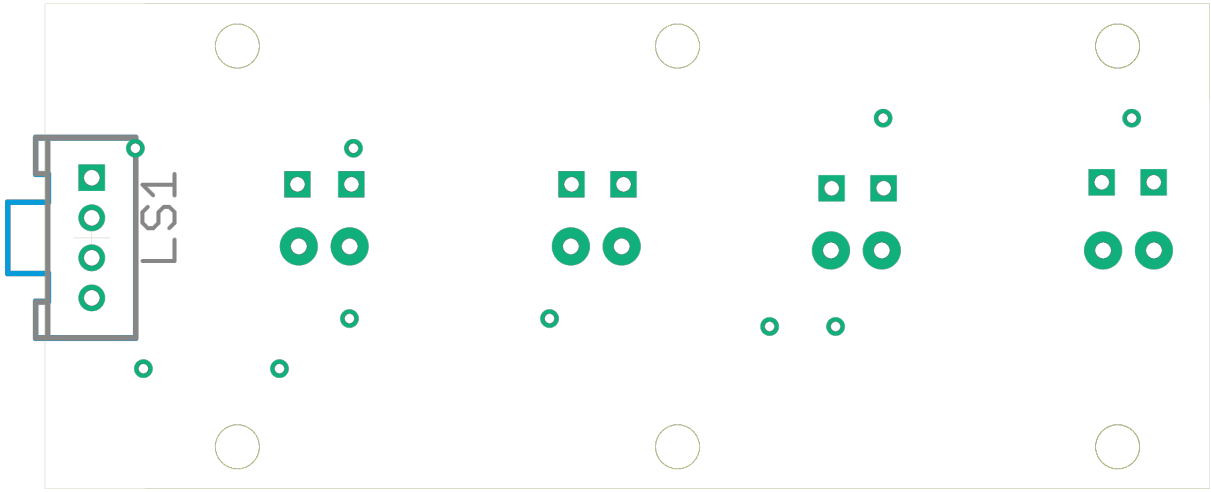
- Bottom



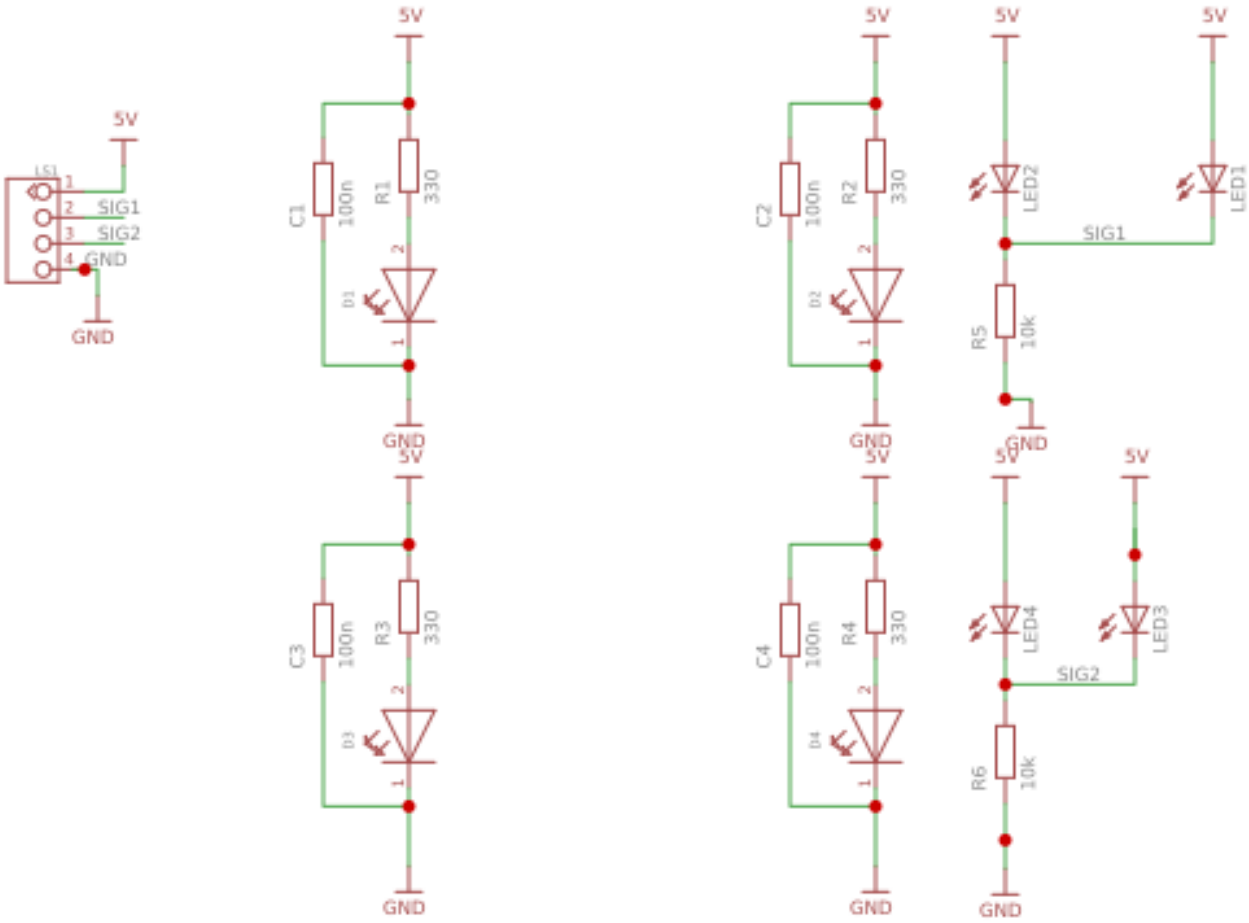
- Components layout top



- Components layout bottom



- Schematic


5V
LS1
1
2
3
4
SIG1
SIG2
GND
GND
5V
C1
100n
R1
330
2
D1
1
GND
5V
C3
100n
R3
330
2
D3
1
GND
5V
C2
100n
R2
330
2
D2
1
GND
5V
LED2
SIG1
R5
10k
GND
5V
LED1
5V
C4
100n
R4
330
2
D4
1
GND
5V
LED4
SIG2
R6
10k
GND
5V
LED3

SOFTWARE DESIGN

Software wise, we designed:

- The Teensy program in C++
- The Arduino Pro Mini program in Arduino C
- The OpenMV program in MicroPython

Every part of the robot, exception made for our camera, programmed in the OpenMV IDE, is programmed via Visual Studio Code paired with PlatformIO, an open source IDE and unified Debugger for many different languages.

The code is fitted for both of our robots to make them as versatile as possible when an Out Of Bounds or an accident occurs, and it's really modular, as it can be easier to comprehend. In this way it's possible to modify the hardware or a specific part of the code without affecting other parts. It's organised in two different ways:

- Many tasks managed by many different functions
- .h to instantiate functions
- .cpp to declare and to write the whole function down
- A "vars.h" to keep track of every variable, included in every part of the code
- A "utility" folder with the old code, datasheets and pinouts

The use of GitHub has helped us with keeeping up with changes as well without a changelog. To understand every feature better many explanatory comments can be found in each file and besides each function and variable. Our code is composed by:

- bluetooth.cpp/.h Bluetooth handling
- camera.cpp/.h Camera routine
- chat.cpp/.h Teammate communication handling

• goalie.cpp/.h	Goalie routine and strategy
• imu.cpp/.h	Imu handling
• keeper.cpp/.h	Keeper routine with the help of the camera
• linesensor.cpp/.h	Line sensor handling and strategy
• main.cpp	Main functions and main variables instatiation
• motors.cpp/.h	Motors and movement handling
• music.cpp/.h	
• nano_ball.cpp/.h	Ball control via slave uC
• pid.cpp/.h	Written down pid for holonomic movement
• position.cpp/.h	ZoneIndex and field position handling
• test.cpp/.h	CLI with tests for every programmable part of the robot
• us.cpp/.h	US sensors handling
• vars.h	Header with every variable used in the code

In the following pages there will be documentation of the most important parts and routines.

HOW DOES IT WORK?

When the robot is turned on, the program starts by completing the setup: this action is announced by four beeps of the buzzer, two for the start and two for the end.
Then, it proceeds by calling the various written routines and strategies, more specifically:

- The Goalie tries to score, and it bases its judgment on camera, ball, lines and US' datas
- The Keeper tries to keep the opponent from scoring, and it bases its judgment on US, lines and ball datas

The robots are bluetooth enabled, this makes them communicate (or "chat") to share information and to let the teammate know if one of them is Out Of Bounds, when the other is not receiving anything.

- Main.cpp flowchart