(Le foto sono state rimosse in modo da poterle scegliere meglio, ci sono un paio di alternative per ogni parte, si trovano tutte nella cartella "pics and pdfs" ed alcune specifiche della parte tecnica e costruttiva in "hardware". Il layout va considerato come ordine degli argomenti e non come impaginazione (è un pochino messy))

Team members

Davide Belli

Age: 17

Hobbies:

Role: Mechanical engineer, chassis designer

Emanuele Coletta

Age: 18

Hobbies:

Role: Software engineer

Siria Sannino

Age: 19

Hobbies:

Role: Software engineer

NEW ROBOTS

(foto robots non in dettaglio)

NG

He is primarily a striker, but he can take part as a keeper as well because of the dynamic role switching. He mounts a BT module to communicate with his teammate and an IMU to know where he is. A new PCB and a new design were crafted for him, so now we can fully use the implemented camera, and we can use four maxon brushless motors instead of three, an efficient body gives him more lightness.

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

(Foto robot 1)

GYU

She is our keeper, but she can take part as a striker as well thanks to the role switch. She mounts a BT module to communicate with her teammate and an IMU to know where he is. A new 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.
(Foto robot 2)

They both feature our new dribbler, to handle the ball better.

THE NEW GENERATION

Our robot works with a single board with everything on it, a big step since 2019's two boards

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:

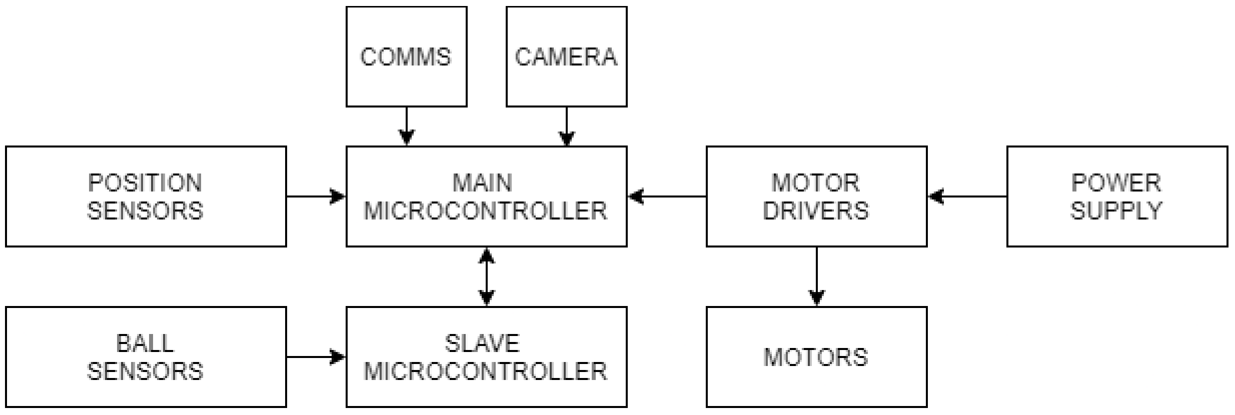
- a new camera with a conic mirror
- single circular PCB for lightness and the absence of connectors
- 16 ball sensors instead of 20, easy to manage and precise
- 4 Maxon motors to move more smoothly and to have more control over the movements
- new line sensors
- new structure and design
- Teensy 3.5 and Arduino Pro Mini
- A dribbler for each robot

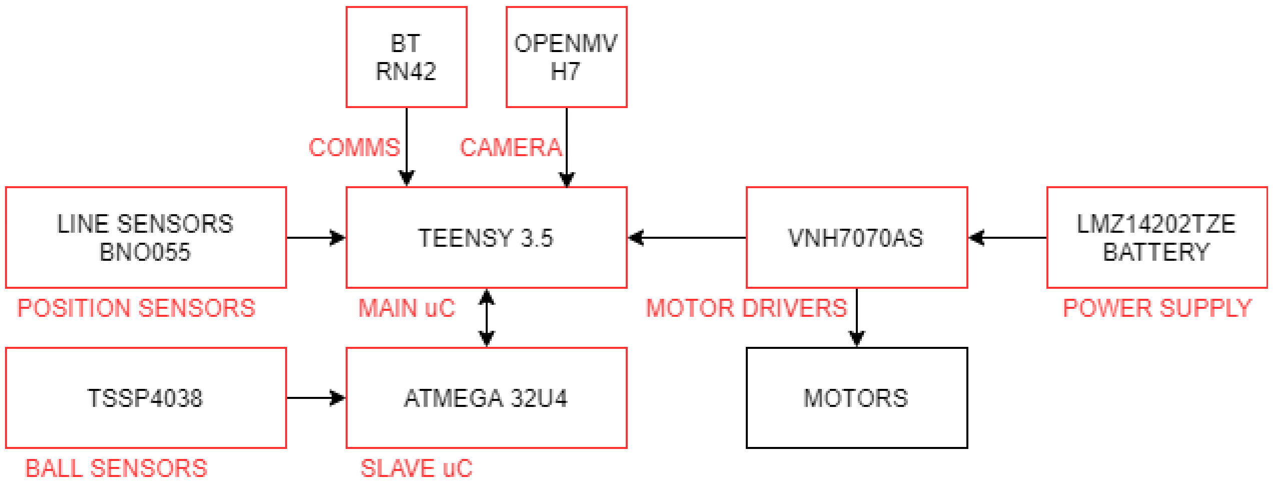
From the software's side, we now have:

- a totally new management of the ball
- a new management of the robot's movement
- a new camera based system to locate ourselves in the field
- a more evolved way to act, using object oriented programming

ROBOT BLOCK DIAGRAM

(da centrare, il file del diagramma è modificabile su draw.io in caso il rosso non vada bene come colore)





WHAT'S ON BOARD?

MICROCONTROLLERS

- Teensy 3.5 (file foto : teensy35 o teensy_1)

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.

- ATMEGA 32U4 (file foto: atmega)

For our slave microcontroller we use an ATMega32U4 in an Arduino Leonardo like layout. We figured it was more compact and efficient than a full board. We use it as a“slave”microcontroller, it manages the ball reading, interpolating the sensors' data and sending it back to the master via UART. It's a low-power 8-bit AVR RISC-based microcontroller featuring 32KB self-programming flash program memory, 2.5KB SRAM, 1KB EEPROM, USB 2.0 full-speed/low speed device, 12-channel 10-bit A/D-converter, and JTAG interface for on-chip-debug. By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching one MIPS per MHz, allowing you to optimize power consumption versus processing speed.

CAMERA (file foto: camera_1 e cam_conic)

- OpenMV H7

The OpenMV H7 Cam is a very powerful, valuable and affordable camera. We mount one of them on each robot, with a conic shaped mirror to see the entire camp and to track the goalposts by color.
It's based on the 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 (file foto: bt_1 o bt_2)

- RN42 BlueSMiRF 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 (file foto : bno055_1 o bno055_2)

- 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.

- Line Sensors

(foto sensori linea nuovi)

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

- Four/six white LEDs
- Four/six phototransistors
- Four/six capacitors
- Four/six 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 (file foto: ball_sens)

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 sensitive 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 (altre foto scheda e foto “regulator”)

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. We now use a single switching voltage regulator which lowers the voltage to 3,3V from 12 without all the unnecessary components we used to have. The LMZ14202TZE was the answer for us, since it's high current resistant, has an embedded inductor and is very small. To work, we paired it with many capacitors for tension stability and every component has its own little voltage filter to guarantee a constant voltage.

DRIVERS & MOTORS

- VNH7070AS Motor Driver (file foto: motordrivers_diagram)

We now use four VNH7070AS motor drivers by ST to drive our motors; this was ultimately the biggest change, since we can now use a single board with everything we need instead of a motor-dedicated one. It's a full H bridge motor driver with no breakout board, but it can also work in half-bridge mode. The input signals INA and INB can directly interface the microcontroller to select the motor direction and the brake condition. A SEL0 pin is available to address the information available on the CS to the microcontroller. The CS diagnostic pin allows to monitor the motor current by delivering a current proportional to the motor current value. The PWM, up to 20 KHz, allows to control the speed of the motor in all possible conditions. In all cases, a low level state on the PWM pin turns off both the LSA and LSB switches.

Features:

- 3 V CMOS-compatible inputs
- Undervoltage shutdown (UVLO)
- Overvoltage clamp
- Thermal shutdown
- Cross-conduction protection
- Current and power limitation, loss of GND and VCC protection

- Motors (file foto: motors,gears e, se c'è spazio, motors_table e gears_table)

Our motors are produced my Maxon, we feature four DCX16L motors with GPX16HP 2-stage gears. The motors have an high power density while being really quiet, small and robust, with a dedicated brush protection (spark suppression – CLL) and ball bearings. Paired with the gears, which feature an extremely high power transmission with a very short construction and a reinforced output, they have made our movement evolve.

- Dribbler (materiale incompleto)

Each one of our robots now has a device called “roller” or “dribbler” to roll the ball in the robot's mouth. This made our striker more powerful and precise. We initially used a brushless DC motor with a 5V ESC, but now we're using a driver and an old brushed DC motor to do the trick and to make it rotate counterclockwise to shoot.

- Holonomic movement (file foto: holonomic)

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 PID controller had to be rewritten to work with the current configuration because their control is different, we now use the ArduinoPIDLibrary.

- 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 an hollow part and small geared discs around the circumference which are perpendicular to the turning direction. So they roll freely in two directions using the rollers mounted around its circumference. 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, so we opted for a change of style and materials.
(foto ruote nuove)

ROBO-BOARDS (foto robot e/o schede non prese dall'alto)

Lots of innovations were brought into our new robots :

- A single, circular board
- We removed the US sensors, which became unreliable
- We placed of a ToF and ball presence sensor connector
- We now use only 4-pin clipped Grove connectors and compact Molex
- We are powered by Maxon motors with new motor drivers
- A completely new supply circuit is on and working

(da qui in ordine mi piacerebbe mettere in ordine queste foto, tutte nella cartella "hardware". Occuperanno una pagina per foglio:

- brd_top
- brd_bottom
- brd_componentlayout_top
- brd_componentlayout_bottom
- (se si ottiene una risoluzione soddisfacente, dato che abbiamo avuto spesso problemi) schematic
- motordrivers
- switches
- supply
- atmega and stuff
- cons_1
- cons_2
- ballsensors

Manca la parte delle PCB dei sensori linea e dei sensori palla presa di cui non ho foto, arriverà nei prossimi giorni e andrà messa in ordine dopo queste foto)

SOFTWARE DESIGN

The software also has a completely new structure. We have three main parts:

- The Teensy program in C++
- The ATMega32U4 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. For the first time we approached a new way of programming, using Object Oriented Programming, this is for a few reasons:

- To organize the code better
- To have more flexibility and plug-and-play capability
- To be more modular and efficient

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 five main folders, in every one of them there are code parts:

- Things with the same names in src
    - behaviour_control
        - complementary_filter.cpp/.h
        - data_source.cpp/.h
        - ds_ctrl.cpp/.h
        - status_vector.cpp/.h
    - motors_movement
        - drivecontroller.cpp/.h
        - motor.cpp
        - roller.cpp
    - sensors
        - data_source_ball.cpp/.h
        - data_source_bno055.cpp/.h
        - data_source_camera_conicmirror.cpp/.h
        - sensors.cpp
    - strategy_roles
        - game.cpp/.h
        - games.cpp/.h
        - keeper.cpp/.h
        - striker.cpp/.h
    - system
        - lines
            - linesys_camera.cpp/.h
            - linesys_camera_recovery.cpp/.h
            - linesys_empty.cpp/.h
        - positions

      - positionsys_camera.cpp/.h
      - positionsys_zone.cpp/.h (old US sensor setup)
      - positionsys_empty.cpp/.h
    - testmenu.cpp/h (test menu doesn't have a folder)
- Other files
  - src
    - main.cpp
  - include
    - vars.h

HOW DOES IT WORK?

Our new code is object oriented, so it uses classes. We have a specific class for everything, from sensors to motors, which inherit from a "master" class with predeclared overridden functions. It also features a "status vector", a data vector which keeps track of important data, so we can now decide what to do based on the comparing of past and present data.

(flowchart mancanti, occuperanno un foglio per grafico)