SPQR Team Description for RoboCupJunior 2021

Coletta Emanuele[1], Sannino Siria[1], Belli Davide[1], Manfrè Massimiliano[1]

[1] ITIS Galileo Galilei, via Conte Verde 51, 00185, Rome, Italy

https://github.com/spqr-team/spqr-team.github.io
manfre.massimiliano@itisgalileiroma.it
spqrrobotics@gmail.com / spqr.roboticteam@itisgalileiroma.it

Abstract. This paper provides a detailed explanation of the mechanical, electrical, and software designs of the SPQR Robotics Team robots in order to compete in the online version of RoboCupJunior 2021. Our robots are made from scratch every single year; even in the unfortunate events that have been happening lately, we've been passionate about our work. The changes from our last RoboCupJunior in 2019 have been substantial to renovate our strategy and looks. On the hardware's side, we now have a single PCB instead of two and a lot of new features. From the software's side, we now use Object Oriented Programming in C++ successfully. Finally, on the mechanical side, we've implemented a conic mirror, a dribbler and lightweight structure, as well as the usage of Maxon motors thanks to the Young Engineers Program.

Keywords: RoboCupJunior, Soccer Lightweight league, SPQR Robotics

1 Introduction

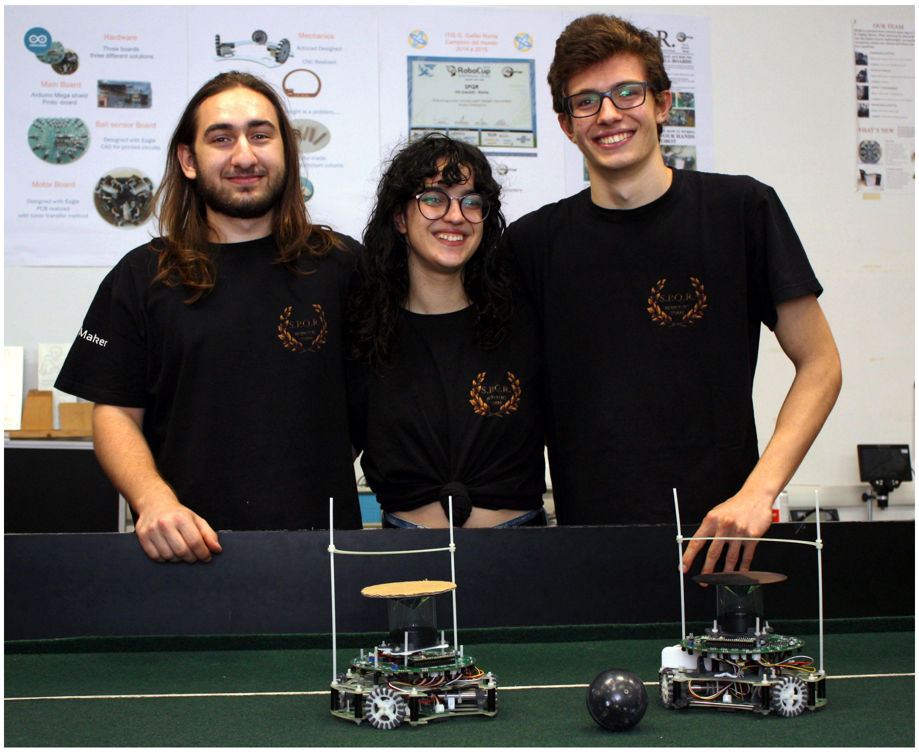
1.1 Team Background

SPQR Robotics Team is a project created in 2009 at ITIS Galileo Galilei of Rome, a technical high school where students come to build robots to compete in the local championships, RomeCup, and RoboCupJunior. We've been taking part in competitions since 2010, we're undefeated on national level; this year, we celebrated our 11th victory in a row with an online version of the RomeCup. Playing in the RoboCup since 2010, our history in the competition is more than positive. We count:

- Third Place in Mexico City, Mexico in 2012
- Best Teamwork in Eindhoven, Netherlands in 2013
- First Place in João Pessoa, Brazil in 2014
- First Place in Hefei, China in 2015
- Best Robot Design in Nagoya, Japan in 2017
- Best Robot Design in Sydney, Australia in 2019

The team for 2021 consists of three people: two software engineers, Emanuele Coletta, the team leader, and Siria Sannino, with a mechanical engineer, Davide Belli.

1.2 Team Photo



1.3 2021 Highlights

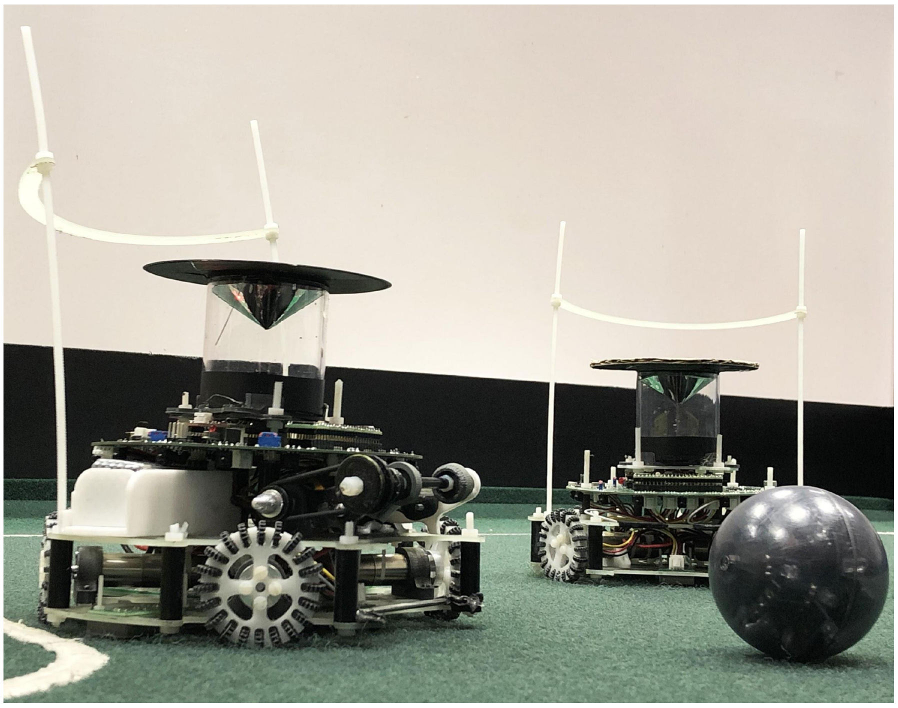
Our robot works with a single board with everything on it, a big step since 2019's two boards. The design was made with Eagle CAD, Fusion and AutoCAD.
Many improvements were made: the hardware is now fully functional and more practical, as the design is now much more efficient and elegant than before. 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
- Onboard ATMega32U4
- A dribbler on one 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

2 Robots and Results



2.1 Hardware

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. 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 a 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.

ATMEGA32U4

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 to manage 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

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.

This camera is also really flexible for all needs, with its OV7725 image sensor, and the lens is unmountable and it can be changed with other ones.

BLUETOOTH

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 pinout as the FTDI Basic, and is meant to plug directly into an Arduino Pro, Pro Mini, or LilyPad Mainboard. To make it 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.

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; 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.

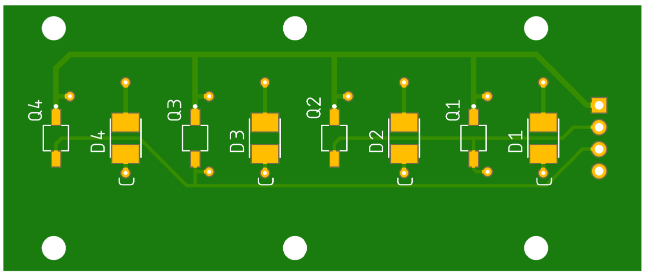
Our software programs the BNO-055 in Mode IMU-PLUS, and it's mounted upside-down to save some PCB space, so we use the P7 configuration. We have a way to control it through a command line interface created by us.

Line Sensors

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.



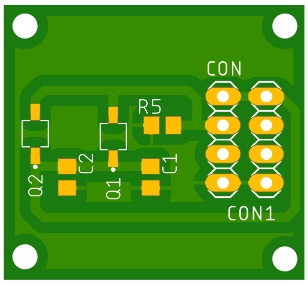
Ball Sensors

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.

Ball Presence sensors

We now use a ball presence sensor composed by two boards, one with two LEDs and the other with two photo-transistors, to know when the ball is in the robot's mouth to interact with the dribbler. We made it from scratch with the photo etching technique.



POWER SUPPLY

This new robot generation's power supply is very different from the past. The main power supply is still the same 3 cells, 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 deal breaker for the robot, since it is 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

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 manages the motor current by delivering a current proportional to the motor current value. The PWM, up to 20 KHz, allows control of 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. They also are very secure, with a UVLO, short and thermal shutdown and current limitation.

Motors

Our motors are produced by Maxon, we feature four DCX16L motors with GPX16HP 2-stage gears. The motors have a 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

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

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 wheel speeds.
Our PID controller had to be rewritten to work with the current configuration because their control is different, we now use the ArduinoPIDLibrary, and we redesigned our wheels to be driven with full force but also to slide laterally with great ease. They are known as omni-wheels, ours have a hollow part and small geared discs around the circumference which are perpendicular to the turning direction.

2.2 Software

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, except 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 keeping 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 method and variable. Code organization has five folders with designated parts within them:

- behavior_control
- motors_movement
- sensors
- strategy_roles
- system

The only parts outside the folders because we need quick access to them are the main, where we have startup actions, the test menu UI, to conduct tests in an easy way and the vars file for frequently used variables.

2.3 Results

Every decision made on the robot was made taking from previous years' experience, especially from the 2019 RoboCup, as we aim to get better and better with each challenge we face. We inherited some functions from it, but deciding to change so much had to be done. Components were picked before starting to construct the robot as an experiment, and software was changed in order to make developers have an easier time with programming, even though the initial work with OOP was the hardest, we can now use more of the potential of the robot.

3 Conclusion and Future Work

In the times of the global pandemic, it was certainly difficult to work in the team, as Italy was one of the most affected countries, so our ideas couldn't come to life. This year, we've learned so much since we started from scratch: for example, the usage of OOP and all the new components. We already have some ideas to be applied in the future: the implementation of a kicker on both of our robots, the change of the structure's material (from fiberglass to aluminum, to ensure lightness and robustness), the change of microcontrollers, for a while we've wanted to remove our slave microcontroller for a code fluidity. We hope to get to see that in future, better times.

References

1. Raul Rojas and Alexander Gloye Föster: Holonomic Control of a robot with an omni-directrional drive (2006).
2. Belli Davide, Coletta Emanuele and Siria Sannino: SPQR Team Repository on GitHub (2020).