compAIR-Autotest

Georg Rohrhofer*, Lukas Haindl, Marcel Schock, Quentin Reisinger
Technical Secondary College
Department of Computer Science
2700 Wiener Neustadt, Austria,
*Corresponding author Email: rohrhofer.georg@student.htlwrn.ac.at

***Abstract*— Due to the COVID-19 pandemic robotics competitions in person got canceled. To create a robotics competition with positive aspects from both, physical and simulated competitions, robo4you created the compAIR [3]. To enable the location independent access to tests under realistic conditions, compAIR-Autotest was created. The system is not only useful in case of the compAIR, but also useful in preparation for ECER. Our system is designed in a way where there are very little limitations to the fields of application.**

I. Introduction

In order to optimize training exercise for robotics competitions and enable digital access to the laboratory from everywhere, we designed a solution to enable remote testing. This system was developed to allow the testing of robots from home at any time. There is a comparable solution called ”RoBox” [2], which enables ROS-based robots to be simulated and also physically started over the internet. This system is not practical in our case, because every participant has to manually select which robot he wants to use, which means the teams always have to check if a robot is available before testing. Our solution is mainly designed the compAIR-Robots. The compAIR is a robotics competition which was created by robo4you [5], where all teams use the same robots during the competition. There is also a publication about implementing a remote robotics laboratory called ”RELLE [1]”, which enables users to program and watch their robots from home. Our solution differs to the existing, by enabling users the access to the robots, even when they are all occupied.

II. Concept

A. Components

Our System uses the following Components:

- website to upload code
  - The website is called compUpload and it provides the users with the necessary access
- queue
  - This queue is used, so multiple test can be started while the robot is running
- software to decide which robot works next
  - This software component assigns robot their next task, based on the ones in the queue
- software to run code in docker [8] container
  - There has to be a component running on each robot, which is used to download the users code and to run it inside a docker container.
- web-server
  - This server is used to host compUpload
- robots
  - These are the robots which are set up to work with compAIR-Autotest.
- server for monitoring robots
  - In order to know which robots are currently in use, our system needs to monitor all robots at all time
- database
  - To recover the robot states after a system-crash we save all states in a database The database is used to store the state of all robots at all time.

When using our system, the provider needs a web-server to host our website where the participants can upload their code, start the test and view the test results.

B. General

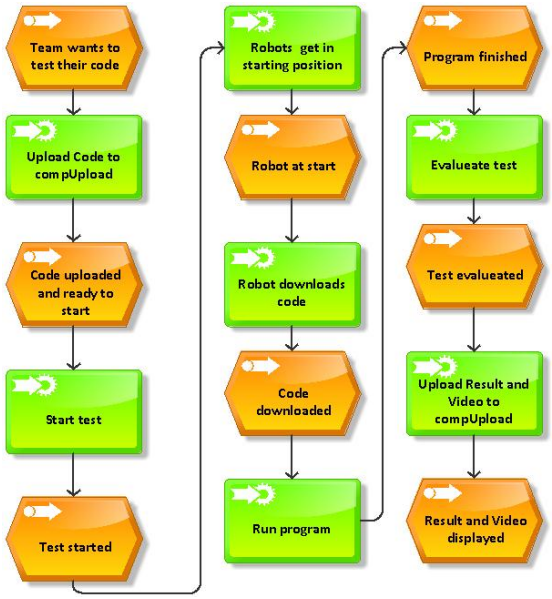

Figure 1. Modeled process for testing

The process, shown in Figure 1, despicts how compAIR-autotest operates. Teams need to program their robots on their local machines. When the teams are ready to test their robots, they need to upload their code to the upload-platform ”compUpload”. When the upload has finished they can now start their test run. Now the robot drives to its starting position and downloads the teams code. When everything is ready the system automatically starts the program. While the robot is driving on the game table a camera, which views the game from on top, captures the whole run. After the program finished or the time ran out, compAIR-autotest evaluates the test run. Finally our system publishes the results and the video. Now the teams can see the test run or compare their result to past runs. If the user wants to watch the video of their run they can click on the desired run, which opens a new tab where they can watch the run. The output of their code is simultaneously simulated next to the video, which means the output of the users program is shown at the moment the users program made the output in comparison to the video.

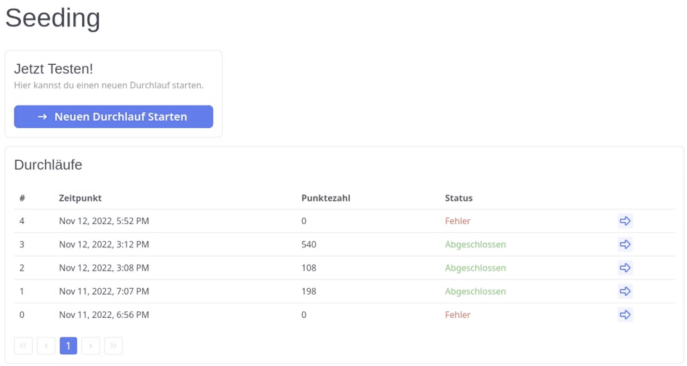

Figure 2. Upload platform statistics

The participants can see on our website, when they did which test-runs as shown in Figure 2. Through that information teams can not only track their progress, but also view back older runs if they want to see how it worked without testing it again.

C. Upload-platform

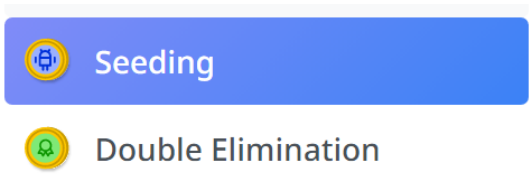

Figure 3. Upload platform game mode

Every team gets its own account, which can be used identify the individual tests. When accessing the website they can choose if they want to upload their code for seeding or double elimination as shown in Figure 3. If they choose double elimination a second robot with either predefined code or with code from another team will be started.

D. Allocation of a test

In order to provide the teams with a queue, where they can schedule a new test-run, the ”RabbitMQ” is used. RabbitMQ is a message broker which allows us to send the tasks over the network, so the upload-platform and the software to decide which robot the code is tested on can run on different machines without the need of manual scaling. The RabbitMQ also provides also a queue for the messages send over the network. This queue is what we use to enable the users not to worry about a free robot to test their code. RabbitMQ is useful in this particular project because it guaranties that messages arrive at the target location. If a message cannot be received the message returns to the queue until the receive was acknowledged. We use this feature to put messages back in the queue when no robot is available for testing. RabbitMQ provides also a layer of security for the messages sent over the network. Access to the queue is controlled with a username and password. We use this security, because otherwise we would need to provide the security for the messages sent.

III. Field of Application

Our system does not only target the current compAIR-robots. Due to plans for changing the robots in the near future our system was developed in a way where there are few hardware limitations. Our system runs on nearly all robot-controllers which are based on the Raspberry Pi [5] or similar systems. Controllers without a network connection or without the ability to run multiple Programs at once are not compatible with compAIR-Autotest. For the different robots only the program to drive back to the starting position has to be swapped.

A. compAIR

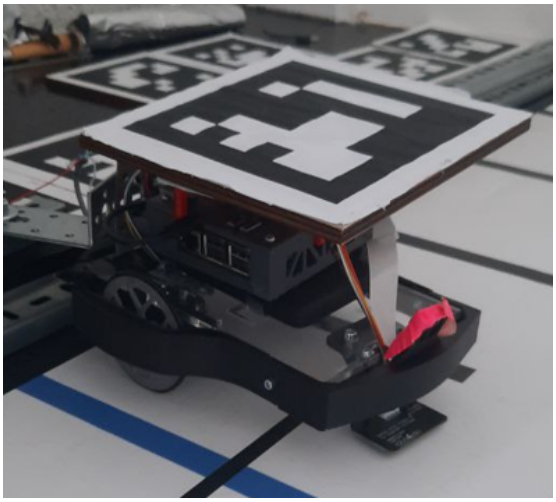

Figure 4. compAIR robot with ArUco [7] Marker

This system is mainly developed for the compAIR, which uses small robots based on the Raspberry Pi as shown in Figure 4. The current robots show weaknesses in their physical design, where the wheels wear down extremely fast and need

to be swapped regularly. If the wheels fail before they get swapped. No new tasks are allowed to be assigned to the robot. To stop the allocation of tests to the failed robot a second program, which monitors the robot is used. Due to the swap of the robots in the next season, this program is not integrated in the main testing program and is only used this season. This workaround receives a copy of the code to test and monitors the movements over the camera which is also used for the video. If the movement shown on video differentiates a lot to the movement suggested by the code, the system stops the running program and blocks further test allocation. The robot gets also marked on the website for the administrators, to show that the robot needs maintenance before it is manually unlocked.

B. Other robots

Our system can be applied when the robot, supports Python, docker and a network connection. The code for the robot, doesn't have to be written in Python. If the robot uses another programming language the only thing that has to be changed is the docker container. The system is implemented in a way where the individual components are independent from each other. If other robots than the compAIR-robots are used the procedure for driving back to the starting position has to be changed. This procedure is the only one accessing the movement of the robot and can not be further generalized.

IV. Value for Botball

A. Changes to Botball

In order to enable to user to run multiple tests, a few changes have to be made to Botball. The issue with the use of our system for Botball is that after each run the game-pieces are in a different state then they used to. In order to grantee the same initial state each time the test is started the game-pieces have to be simulated. The compAIR has already proven, that a robotics competition with simulated game-pieces can work.

To simulate the game-pieces, the game-table has ArUco-markers placed around it, so our system can calculate the initial positions for the game-pieces. In order to track the pieces when the robots move them to a different position during the game, the robots are continuously tracked during the test by multiple cameras. Due to the simulated game-pieces our system also offers a fully automatic test evaluation.

In Botball not all robots use the same hardware. In this situation provide the teams with a software where they can upload a 3D-model of their robot. On this model, locations where ArUcO-markers are placed, need to be defined. These markers then get also physically mounted to the Robots. Our system is now able to track the robots even if they hardware used varies a lot.

B. Preparation for ECER

Our system can also be used in the preparation phase for a Botball tournament such as ECER [6]. In this case compAIR-Autotest can be used to test code as soon as the construction of the robots is finished. In order to enable automatic reset of the position of the robot, an ArUco-Marker has to be placed on top of the robots. To enable teams to use our system in preparation for ECER the robots need to get linked directly to the teams user account. Also each team has to setup the starting position for both robots and save the position, so compAIR-Autotest can reset the starting position after a run has finished. If a team wants to test new code, they need to upload their code to the upload platform and select if the code is supposed to run on their main or second robot. If the team is ready they can start the run.

Our system is also advantageous while taking part in a Botball competition. Due to our system monitoring all robots at any time, it's easier to track progress in competition and fix errors in a short time. Our system offers an automatic statistics generation. This enables users to see if their new features have a positive impact on their strategy or if they should change it. Due to the automatically testing evaluation the scores get published on the website. These scores are then displayed in a chart. The scores can also be marked, which robot achieved them so more specific statistics can be generated.

C. Course of training

We modeled the process of training divided into setting up the robots to be used with compAIR-Autotest and the process of the training itself.

1) Setting up the robots: In order to use our system the teams first need to set their robots up, so they can communicate with our software. Both of the teams robots need an ArUco Marker to locate their position on the table simular to the compAIR-Robot shown in Figure 4. When the robots get set up, they need to be set to their starting position. Our system can then save this starting position automatically for each robot. Our software also needs to be installed on every robot, that should be used with our system. When the robot is first used with compAIR-Autotest the robot needs to be specified inside our database, so our system knows which robots are available.

2) Training: If a team wants t o test their code they start a new run. After the test run our system fully automatically evaluates the test. The system calculates the achieved points based on the scoring rules of the year. After the run is evaluated the points, the code output and the captured video are shown on the upload platform. There is also different statistics displayed on the upload platform. Shown statistics are the change in points in different runs, the average points and, to ease up the preparation phase and to visualize the consistency, the standard deviation.

With our current implementation our system can run python code. When a user wants to test their code, they has to name the main file "main.py", so our system can start it without asking which one is the main file. All of the source files need to be compressed in a .zip file. Our robots then automatically download the .zip file and run it. To enable our system to execute also programs written in another programming language, only the dockerfile, with contains the commands to start the users code, needs to be swapped out. The dockerfile needs to be changed on every robot, where the teams want a different programming language. Due to the independent architecture of our system in regard of the programming language used, not all robots need to be programmed in the same programming language.

V. Why use our solution

In order to provide a base for comparison with competition systems, this section is a SWOT-Analysis.

A. Strengths

Our system provides location independent access to the robots in the laboratory. compAIR-autotest offers not only a access from everywhere in the world, but also the possibility for users to start their test run, even if there are no free robots.

B. Weaknesses

In order to fully fulfill the main use-cases of our system there has to be a lot of teams trying to test their code at the same time. Due to the main field of application in the compAIR where there are only a few robots for a lot of teams, our system is designed to optimized the user experience in comparison to other solutions, like simply connecting to the robots over ssh.

C. Opportunities

Our solution offers robo4you to expand the compAIR and keep costs down. Our system needs only enough robots for each game mode to function properly. In this case compAIR-Autotest is necessary, to enable all users the ability to test. If this system is not used, the robots could be blocked by the same teams for a long time, so others can't test their code. Our system also offers a protection layer for the robots, so teams can't manipulate important aspects of the robots operating system.

D. Threats

Our system does not make a lot of sense if there is only one team accessing the robots at a given time, because the cost of deployment is higher than the value in return. In these cases solutions which enable direct access to robots are more effective than ours.

VI. Conclusion

The COVID-19 pandemic showed, how easily robotics competition are prevented from taking place. For this reason the compAIR got developed. Due to it's design it allows the users to take part at a robotics competition. Our system is now the next step to enable a real competition without the need of human judges. This not only prevents the spread of deceases in the future, but also eliminates the potential user error by the judges. In our case we need this system to enable a lot more teams, to test their code in an environment, that resembles the same condition as the one used during the competition. Our system allows also for more dramatic changes to the game-environment, because teams don't need to invest the extra cost of buying or modifying the existing game table from previous years. Our system enables also a lower entry barrier for the teams to be a part of the compAIR, because they don't need to buy a robot, in order to test their code.

References

[1] de Lima, J.P.C., Carlos, L.M., Schardosim Sim˜ao, J.P., Pereira, J., Mafra, P.M., and da Silva, J.B. (2016). "Design and implementation of a remote lab for teaching programming and robotics.", IFAC-PapersOnLine, 49(30), 86–91. doi:10.1016/j.ifacol.2016.11.133. 4th IFAC Symposium on Telematics Applications TA 2016.

[2] The Construct, "RoBox - 24/7 ROS Remote Real Robot Labs" "https://www.theconstructsim.com/robox", 2022

[3] Verein zur Förderung von Wissenschaft und Technik an Schulen (F-WuTS), "Competitive Robotics Reimagined," https://comp-air.at, 2022

[4] Verein zur Förderung von Wissenschaft und Technik an Schulen (F-WuTS), "robo4you", https://robo4you.at, 2022

[5] A. Nayyar and V. Puri, "Raspberry Pi-A Small , Powerful, Cost Effective and Efficient Form Factor Computer: A Review," Int. J. Adv. Res. Comput. Sci. Softw. Eng. 5(12), vol. 5, no. 12, pp. 720–737, 2015

[6] PRIA, "ECER 2023", https://ecer.pria.at, 2023

[7] Babinec, Andrej & Jurišica, Ladislav & Hubinský, Peter & Duchoň, František, "Visual Localization of Mobile Robot Using Artificial Markers", 96. 10.1016/j.proeng.2014.12.091, 2014

[8] Docker Inc, "Docker: Accelerated, Containerized Application Development", https://www.docker.com, 2023

[9] VMware, Inc "Messaging that just works - RabbitMQ", https://www.rabbitmq.com, 2023