published as: Martin Malchow, Jan Renz, Matthias Bauer, Christoph Meinel: Embedded Smart Home - Remote Lab Grading in a MOOC with over 6000 Participants, In Proc. of the 2017 Annual IEEE Systems Conference (SysCon '17), 24-27 April, 2017, Montreal (Canada)

Embedded Smart Home - Remote Lab Grading in a MOOC with over 6000 Participants

Martin Malchow, Jan Renz, Matthias Bauer, Christoph Meinel
Hasso Plattner Institute (HPI)
University of Potsdam
Potsdam, Germany
Email: {martin.malchow, jan.renz, matthias.bauer, christoph.meinel}@hpi.de

***Abstract*—The popularity of MOOCs has increased considerably in the last years. A typical MOOC course consists of video content, self tests after a video and homework, which is normally in multiple choice format. After solving this homeworks for every week of a MOOC, the final exam certificate can be issued when the student has reached a sufficient score. There are also some attempts to include practical tasks, such as programming, in MOOCs for grading. Nevertheless, until now there is no known possibility to teach embedded system programming in a MOOC course where the programming can be done in a remote lab and where grading of the tasks is additionally possible. This embedded programming includes communication over GPIO pins to control LEDs and measure sensor values. We started a MOOC course called "Embedded Smart Home" as a pilot to prove the concept to teach real hardware programming in a MOOC environment under real life MOOC conditions with over 6000 students. Furthermore, also students with real hardware have the possibility to program on their own real hardware and grade their results in the MOOC course. Finally, we evaluate our approach and analyze the student acceptance of this approach to offer a course on embedded programming. We also analyze the hardware usage and working time of students solving tasks to find out if real hardware programming is an advantage and motivating achievement to support students learning success.**

I. Introduction

MOOCs have enjoyed great popularity for several years now. Usually a MOOC course is discusses theoretical lecture topics. In case of the MOOC platform we used, openHPI[1], IT courses such as Web Technologies, Semantic Web Technologies, or In-Memory Data Management are offered for an interested public. Using an online programming environment extends the possibility in a course and the variety of interactive student tasks. Nevertheless, we faced the problem of interactive programming in real hardware. Such as an embedded system. Therefore, we begin the Embedded Smart Home course as a case study on our MOOC platform, which gives users the possibility to work in a remote lab on the real hardware. The hardware used for this approach was based on the Raspberry Pi[2]. To attract students to participate in this course we decide on a smart home and IoT (Internet of Things) focus. Course participants had the option to use their own hardware at home or to use the remote lab. Both options can grade their results and thereby offer the necessary points for a certificate or proof of participation.

[1]https://open.hpi.de/
[2]https://www.raspberrypi.org/

Finally, the Embedded Smart Home Course has over 6000 enrollments. This high amount of students leads to high traffic on the remote lab. To handle the requests a smart request handling was implemented, which will be described in the section "Approach". Furthermore, besides further course statistics the section "Evaluation" shows that the used approach to handle requests for 6000 enrollments is adequate. Finally, we discuss "Results and Future Work" for the approach and how it can be improved for other scenarios and interactive MOOC courses online.

II. Related Work

An early approach on a virtual remote lab to demonstrate network security issues was the "Cyber Security Virtual Lab" [1]. This approach was using the VNC technology to connect to a virtual sandbox machine to test in a given network technologies security issues. The student get an idea which security risk exists and how to detect them to avoid attacks by enemies. This approach is a self assessment option only and no grading for the solved tasks can be issued.

An early approach on grading practical programming in the MOOC environment is Code Ocean [2]. Code Ocean is connected to a MOOC using Learning Tools Interoperability (LTI). This enables the MOOC platform to redirect to Code Ocean. The student will solve the task in Code Ocean and grade their programming results using programming language specific test frameworks. One major advantage of Code Ocean is that the platform is independent on the programming language.

There are already several concepts on using remote labs in MOOCs [3][4][5]. This concepts already describing the handling of remote labs in MOOCs. Nevertheless, there is a lack of a huge student basis and the system were not designed as course that students can build up their lab also at home. This is also a reason for our decision to use cheap and common hardware like the Raspberry Pi.

As first evaluation we introduced as additional tasks of a "Java" and "Web technologies" the remote programming of an 16x16 LED matrix [6][7]. To control the matrix students programmed in the course programming language (Java and JavaScript) arrays and were able to control the content of the LED matrix in several colors. Due to the overwhelming feedback we decided to offer the Embedded Smart Home course described in the following "Approach" section.

III. Approach

This section describes the MOOC approach of the "Embedded Smart Home" course. The purpose of this course was to teach the basics of the Internet of Things (IoT) and give users the opportunity to implement software for a real device and run this software on a the real device with visual feedback. Additionally, students should have the possibility to buy their own hardware and work on the real device.

A. Course Structure

The "Embedded Smart Home" MOOC workshop running on the openHPI platform was structured in 2 course weeks plus an introduction week and an excursion week. The introduction week started already one week before the course officially began. This week was meant to be an introduction for students using their own hardware at home. In this week we discussed how to install the Raspbian[3] debian based operation system on the Raspberry Pi. After installing the operating system, four videos were recorded describing how to connect all hardware with the Raspberry Pi by use of a bread board. This bread board connections are shown in Figure 1. The first video describes the connection of the Buttons and LEDs. Followed by the second video showing the connection of the analog to digital converter (ADC) , the humidity sensor and the temperature sensor. The third video describes how to connect the 16x2 LCD display module. The last bread board video shows connecting instructions for the outside temperature sensor and the window switch. About 700 of the 6000 enrolled students decided to buy the real hardware to interactively build the hardware parallel to the course. As an additional hardware option we offered students the possibility to buy an PCB (printed circuit board) shown in Figure 2 which eases the process of connecting the Raspberry Pi with the hardware, as compared to the previously described bread board version. More than 400 students decided to buy the additional PCB for this course to reduce the wiring complexity.

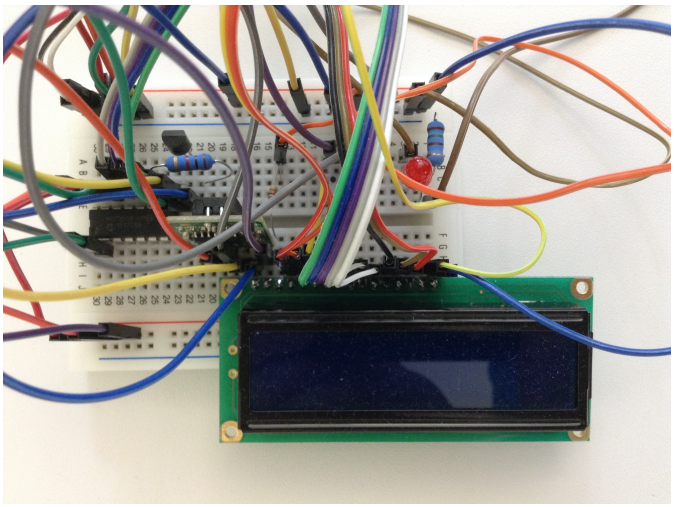

Fig. 1. Raspberry Pi connection with bread board.

After the introduction the course starts with Week One. The first week starts with basic information about the main

[3]https://www.raspbian.org/

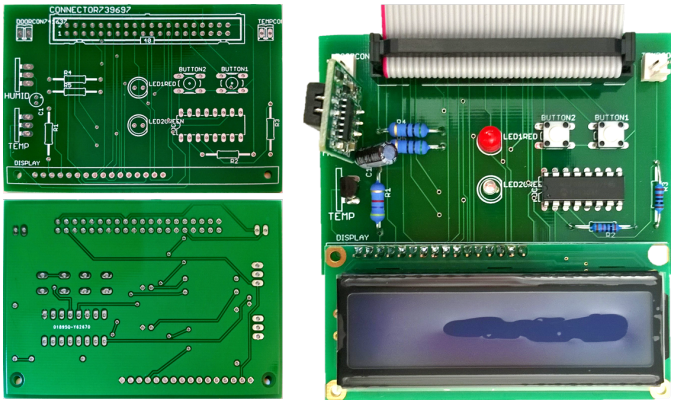

Fig. 2. PCB embedded smart home.

topic "Embedded Smart Home". First the term "Internet of Things" was introduced and use cases of this technology were discussed. This was followed by a video about "Smart Home" in general and the use of "Raspberry Pi". After this short, general introduction we start focusing on programming. Since openHPI had already offered an Python[4] MOOC around half a year earlier, and Python is also used for programming the Raspberry Pi we decided to refer to this course if students needed more training in Python. We informed the students already a couple weeks that they should use this course to refresh their Python knowledge to be ready for the "Embedded Smart Home" course. For Programming and automatic code checks and grading we used the open source software Code Ocean[5], developed by the openHPI Team. In Week One the usage of the Code Ocean platform was described. Additionally we describe how to read a button status, a sensor value by use of an analog to digital converter (ADC), and how to display data on the 16x2 LCD display. The interactive programming tasks where users got points for a certificate are discussed in Section III-B.

After introducing all basics for programming the Raspberry Pi in the "Embedded Smart Home" context with the selected hardware, Week Two focus on the entire system in observing the home to achieve a healthy indoor climate. To reach this goal all sensors, LEDs, and the LCD display have to be observed constantly. All necessary components are visualized in Figure 3. Additionally, in the second week the functionality of a web server and the basic creation of a web server in Python was explained to offer all measured data for an external observing service or interactive smartphone apps. This data will be used mostly for services described in the last excursion week. The LEDs will be used to indicate if a window action is necessary. A red LED indicates that an user action is required and a green LED indicates everything is fine and no action is necessary. First of all the temperature sensor has to be analyzed. Depending on the current indoor and outdoor temperature the window should be closed or opened. Closing of the window is necessary when the temperature is between 10-15 °C and if the window is open for more than 20 minutes this will be indicated by a red LED. The time frame changes depend on the outside temperature. For example when the temperature is below -5 °C the red LED will switch on when

[4]https://www.python.org/
[5]https://github.com/openHPI/codeocean

TABLE I. INDOOR CLIMATE HUMIDITY RECOMMENDATIONS

Cold Weather (below 5 °C)	
22-24 °C	30-40% relative humidity
19-21 °C	40-50% relative humidity
16-18 °C	50-60% relative humidity
Mild Weather (5-15 °CC)	
22-24 °C	40-50% relative humidity
19-21 °C	50-60% relative humidity
16-18 °C	60-70% relative humidity

the window is open for more than 3 minutes indicating that the window should be closed. There are also indicators to show that it is necessary to open the window. Especially in winter the humidity should meet special ranges for an healthy indoor climate. The recommended relative humidity range for different temperatures can be found in Table I. The main programming task was to perform actions with the hardware components to achieve an healthy indoor climate.

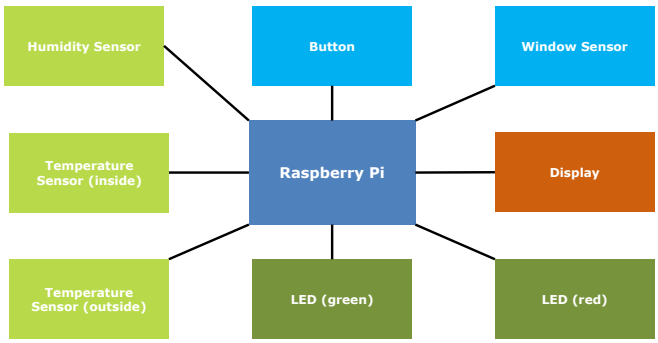

Fig. 3. Abstract connections.

Finally the "Excursion Week" offered a deeper view into the world of the smart home. First the future of smart home and security issues were discussed. Students who bought real hardware got new ideas about how to enhance their system with online services. In this particular course we were focusing on openHAB[6] which enables users to collect data from the embedded device on a server and make them accessible over the web and by Android and iOS App.

B. Programming Tasks

During a MOOC course students have the possibility to get a certificate depending on their achievement. To reach points qualifying the student for a certificate the theoretical homework questions in Week One and Two must be answered. Additionally, the Python programming tasks can be solved at home or at the distance lab described in the following Section III-C. After solving the tasks at home or inside the distance lab, Python unit tests will grade students code in Code Ocean. This unit test grading process will calculate points that are transferred from Code Ocean to the actual MOOC platform openHPI.

The programming tasks in week one are:

- Task 1.1 - Write an application that turns on the red LED.
- Task 1.2 - Write an application that switches the red LED on and the green LED off when the window is closed. When the window is open the green LED should be on and the red LED off.
- Task 1.3 - Write an application that read the temperature sensor value from the ADC and calculate with the given information the applied voltage and the resultant temperature in degree Celsius. Write the result on the command line in the given format.
- Task 1.4 - Write an application that read the humidity sensor value from the ADC and calculate with the given information the applied voltage and the resultant relative humidity. Keep in mind that the temperature is necessary to determine the relative humidity. Write the result on the command line in the given format.
- Task 1.5 - Read the given functions carefully. Write an application that writes "Hallo openHPI!" on the LCD display. The space should be interpreted as a new line.
- Task 1.6 - Write an application that writes the actual room temperature in the given format on the LCD display.

Fig. 4. Display state solving Task 2.3.

The programming tasks in week two are more complex and depend on several sensor values, the last task also introduced the JSON exchange format to connect external services:

- Task 2.1 - Write an application that measures the humidity and window state. If the window is open the green LED is on and the red LED is off. The same LED indication is shown when the window is closed and the relative humidity is below or equals 50% relative humidity. When the humidity is higher, the red LED should be on and the green LED off indicating that opening the window is required.
- Task 2.2 - Write an application that extend the accuracy of the previous task by use of the recommended room humidity displayed in Table I.
- Task 2.3 - Write an application that extends the functionality of the previous task by use of the LCD

[6]http://www.openhab.org/

display. In the first display line print the indoor temperature, the relative humidity, and the outdoor temperature. Additionally, print in the second line “open window”, when necessary. A possible running solution is visible in Figure 4.

- Task 2.4 - Write an application that extends the functionality of the previous task by use of the outdoor sensor to decide if closing a window is necessary. The window should be closed when it is open and the outdoor temperature is below 18 °C. Print the message “close window” on the second display line when necessary.
- Task 2.5 - Write an application that reads all sensor data and print it JSON formatted in the console as shown in the example.

C. Experiment Setup

1) Circuit Layout: As already described in Section III-A we offered a PCB to the students for purchase. This board shown in Figure 2 is also used for our remote lab. We decided on this board since it is more stable and robust for use in the remote lab. This remote device is available in our office so that we can actually see the activities of the students. To design this board we used the Software eagle[7]. The circuit layout showing all connected hardware with the 40 pin connector of the Raspberry Pi is visualized in Figure 5.

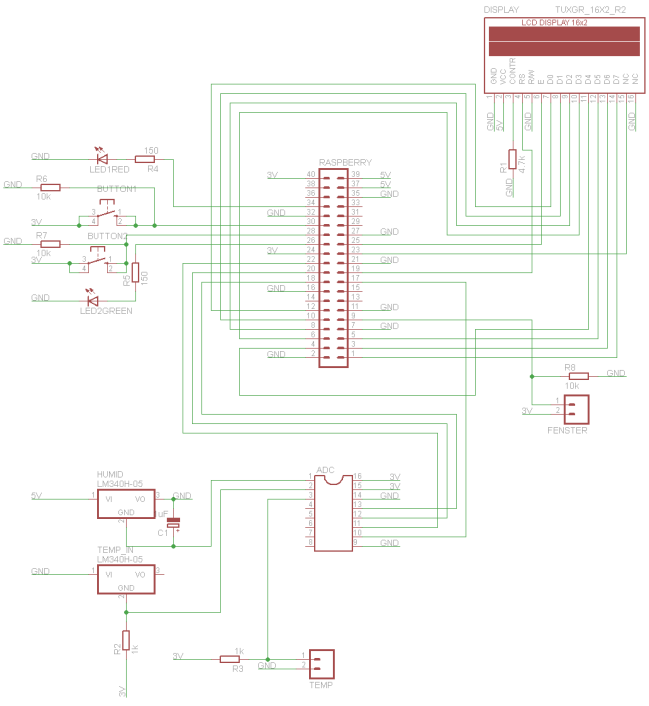

Fig. 5. Circuit embedded smart home.

After PCB assembly we soldered all electrical parts and connectors for external hardware. External hardware like the outdoor temperature sensor or Raspberry Pi is connected to the board with a cable and the fitting connector. After mounting all parts on the PCB the board is ready to use for personal usage or can be used in the virtual lab. For the virtual lab additional steps are necessary. These are described in the following section.

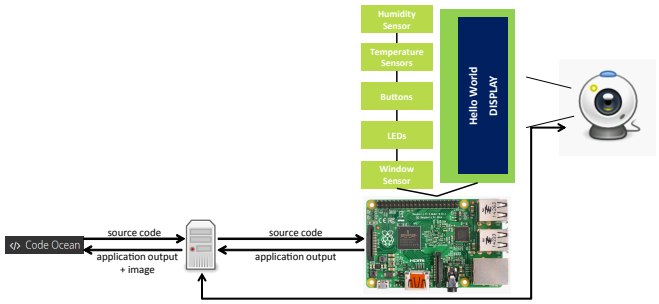

Fig. 6. Remote lab set up schematics.

2) Setup Remote Lab: The remote lab is accessible during the course by the already mentioned platform Code Ocean. The detailed execution process to run an application in the remote lab is shown in Figure 6. In the first step Code Ocean collects all programming files of the student and sends the file content in a special JSON format to the embedded system management server. This server collects the JSON data and checks the availability of the Raspberry Pi using a critical section to avoid multiple access of the embedded device by several users. If several users want to access the Raspberry Pi the critical section can reached by one user only. The other user has to wait until the first user leaves this critical section. The waiting time normally is recognized as a longer request time by the users. Since one user uses the system for a maximal time of five seconds, the waiting time should be reasonable. The execution time for the user is also limited to five seconds so that the system is not constantly blocked when a students construct an endless loop. When a student requests access to a critical section a connection to the Raspberry Pi will be established. This established connection will be used to sent the files JSON formatted to the Raspberry Pi. The Raspberry Pi extracts the file content from the JSON string and writes this files to the storage of the Raspberry Pi. The JSON string also contains information about the main file which will be executed with python in the next step. This will start the application remotely on the Raspberry Pi. Finally, the application output will be sent back over the embedded system management server and Code Ocean to the student. Additionally, the embedded system management server shoots an image of the Raspberry Pi, the LCD display, and the LEDs. This image will be attached to the answer for the student and will be displayed in Code Ocean. The images gives the student visual feedback and programming errors will be highlighted by the returned console output containing standard output and error output. After every request to the Raspberry Pi a clean up of the GPIO Pins will be performed and user files will be deleted.

IV. Evaluation

In this section we will discuss the evaluation results of the hardware availability in the course, learners engagement, and learners feedback to the course.

[7]https://cadsoft.io/

A. Availability

When providing a remote lab in an salable learning environment one of the major issues is that the scalebility of the overall learning journey should be kept all the time. While virtual solutions can be scaled easily by providing additional IT resources a remote lab must be accessable all the time with a very low to now waiting time to match the learners experience. They might be cases where a lab is booked for interactive time slots where a waiting time or a prebooking of time slots is acceptable, but in our case the interaction of the learner with the system is based on a long time-span in the code editor and a very short code execution within the remote lab environment.

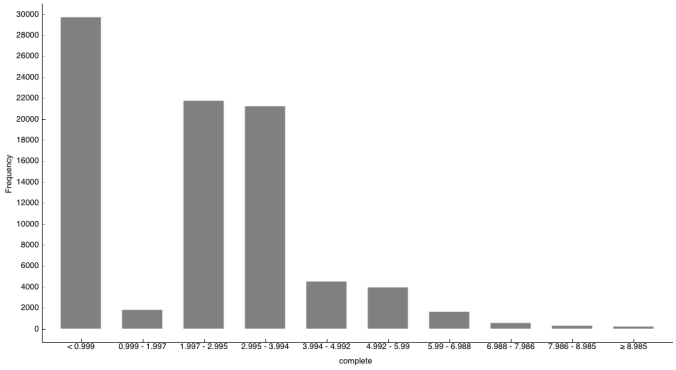

Fig. 7. Response time of the remote lab

86,598 executions have been processed. 34.5 % of all executions have been processed within one second. 8.6 % took longer than five seconds. 51.808 executions including the usage of the camera module, which an average an time of 2.5 seconds. So while a majority of the requests could be performed within an short timeframe there is room for improvement. This includes optimizing the process of taking the picture itself which could be improved by using a video stream instead of taking single pictures.

B. Learners Engagement

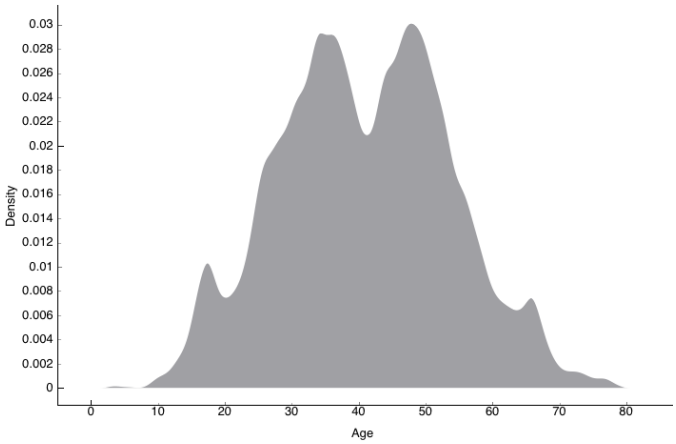

Fig. 8. Age distribution of learners

5.704 learners have been enrolled for the course during course middle. 2274 users never showed up after the course started ("no-shows"). 330 questions have been asked in the forum, 384 answers have been given. Over 1,000 comments have been posted in the forum. 101 helpdesk issues have been opened. At the end of the course 602 certificates have been issued, which leads to an success rate of 11.08 %. This rate is relative low compared to other hands-on coding courses. This might be due to the fact that many users wanted to play around on own hardware that was sold out. However as shown in the item discovery chart many learners discovered large parts of the course, so content consumption was good.

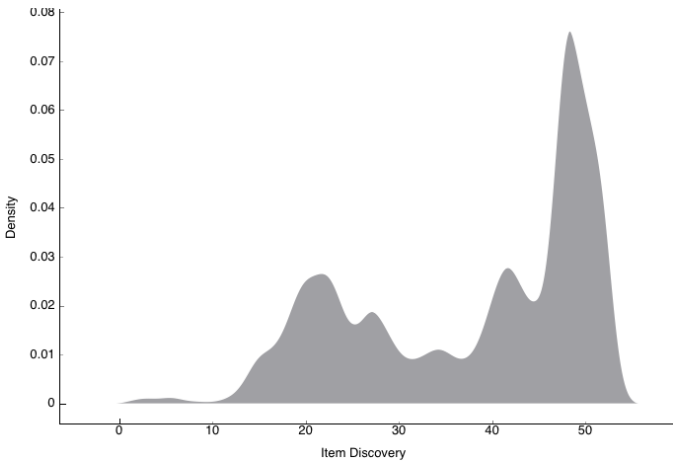

Fig. 9. Item discovery

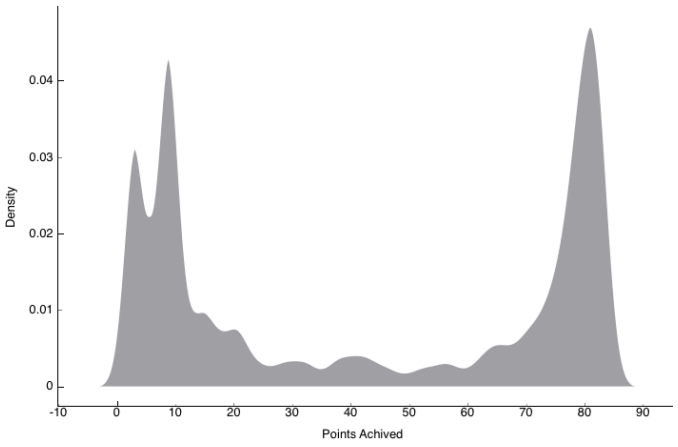

Fig. 10. Points distribution

The points distribution show an uncommon peak in the lower part. This might be due to users who just took the graded tests or users who just took some online coding tasks.

C. Learners Feedback

93 % of the users that submitted the final survey stated (n: 158) that they would recommend the course. Another interesting insight from the course survey is the fact that 71 % of the learners downloaded the videos to use them offline or outside of the mooc enviroment.

During the course we got a lot of positive user feedback and several users would like to have a follow up course with Raspberry Pi. This feedback indicated that we are on

a good way to improve students motivation and can support the learning process. Here is an extract of the translated users course feedback:

- I like the opportunity to control a Raspberry Pi in Potsdam over the internet. To see a real picture as result especially delight me.
- THANK YOU openHPI team please go on. I would love to see new courses which extends the Raspberry Pi topic.
- Code Ocean is awesome :-) I really liked the live programming of a online RasPi. Thank you for this great infrastructure!!! :-)

V. Results and Future Work

In this paper we focusing on the approach of a MOOC remote lab using an Raspberry Pi. This approach shows that we can handle over 6000 enrolled students in a MOOC course on a single Embedded Device. Like described in Section IV-A only 8.6 % of the requests had waiting times over 5 seconds. This should be improved for a possible follow up course by introducing an interactive load balancing on one or more embedded devices. Furthermore, the course design and idea of the real hardware MOOC in a remote lab and at home for the students was luckily chosen. 93 % of the students would recommend this course. Nevertheless, in the next course we have to find a better solution with more than one retailer for the hardware. Due to the sold out situation some people were unhappy that they do not have the chance to order their own device. Additionally, we have to analyze how younger students could be delight for this play full topic. Currently, most students are 30 and 50 like visualized in Figure 8.

In conclusion this MOOC course is well designed using the remote lab. There are still some drawbacks which can be handled in further courses. The feedback and evaluation indicated that the design of this approach is useful and more MOOC courses teaching programming should focus on real hardware.

References

[1] C. Willems and C. Meinel, "Online assessment for hands-on cyber security training in a virtual lab," in *Global Engineering Education Conference (EDUCON), 2012 IEEE*, April 2012, pp. 1–10.

[2] T. Staubitz, H. Klement, R. Teusner, J. Renz, and C. Meinel, "Codeocean - a versatile platform for practical programming excercises in online environments," in *2016 IEEE Global Engineering Education Conference (EDUCON)*, April 2016, pp. 314–323.

[3] T. R. Ortelt, S. Pekasch, K. Lensing, P. J. Guï£¡no, D. May, and A. E. Tekkaya, "Concepts of the international manufacturing remote lab (mintrelab): Combination of a mooc and a remote lab for a manufacturing technology online course," in *2016 IEEE Global Engineering Education Conference (EDUCON)*, April 2016, pp. 602–607.

[4] G. Dï£¡az, F. G. Loro, M. Castro, M. Tawfik, E. Sancristobal, and S. Monteso, "Remote electronics lab within a mooc: Design and preliminary results," in *2013 2nd Experiment@ International Conference (exp.at'13)*, Sept 2013, pp. 89–93.

[5] C. Salzmann, D. Gillet, and Y. Piguet, "Mools for moocs: A first edx scalable implementation," in *2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, Feb 2016, pp. 246–251.

[6] M. Malchow, J. Renz, M. Bauer, and C. Meinel, "Improved e-learning experience with embedded led system," in *2016 Annual IEEE Systems Conference (SysCon)*, April 2016, pp. 1–6.

[7] ——, "Enhance embedded system e-leaming experience with sensors," in *2016 IEEE Global Engineering Education Conference (EDUCON)*, April 2016, pp. 175–183.