published as: Martin Malchow, Jan Renz, Matthias Bauer, Christoph Meinel: Enhance Embedded System E-learning Experience with Sensors, In Proc. of the 2016 IEEE Global Engineering Education Conference (EDUCON '16), 10-13 April, 2016, Abu Dhabi (UAE)

Enhance Embedded System E-learning Experience with Sensors

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

***Abstract*—Earlier research shows that using an embedded LED system motivates students to learn programming languages in massive open online courses (MOOCs) efficiently. Since this earlier approach was very successful the system should be improved to increase the learning experience for students during programming exercises. The problem of the current system is that only a static image was shown on the LED matrix controlled by students' array programming over the embedded system. The idea of this paper is to change this static behavior into a dynamic display of information on the LED matrix by the use of sensors which are connected with the embedded system. For this approach a light sensor and a temperature sensor are connected to an analog-to-digital converter (ADC) port of the embedded system. These sensors' values can be read by the students to compute the correct output for the LED matrix. The result is captured and sent back to the students for direct feedback. Furthermore, unit tests can be used to automatically evaluate the programming results. The system was evaluated during a MOOC course about Web Technologies using JavaScript. Evaluation results are taken from the student's feedback and an evaluation of the students' code executions on the system. The positive feedback and the evaluation of the students' executions, which shows a higher amount of code executions compared to standard programming tasks and the fact that students solving these tasks have overall better course results, highlight the advantage of the approach. Due to the evaluation results, this approach should be used in e-learning e.g. MOOCs teaching programming languages to increase the learning experience and motivate students to learn programming.**

Keywords—Teleteaching, Tele-Lecturing, Distance Learning, E-Learning, Virtual Lab

I. INTRODUCTION

Motivating students in Massive Open Online Courses (MOOCs) is a difficult task. Normally, MOOC students have other life obligations like a job or family. Therefore, some additional motivation is needed to actively participate in MOOC courses. An interesting idea in MOOCs dealing with learning programming languages or using programming languages to emphasize examples is the use of an interactive embedded system, which can be accessed and evaluated through the MOOC platform [1]. This embedded system can display data on an LED display by submitted user arrays over the Code Ocean platform by several programming languages. However, this approach is somewhat static and the embedded system can only receive data. This paper describes the idea to improve the system. Therefore, the embedded system should get a dynamic component to avoid static array creation to show an image on the LED display to solve the tasks. This means that the embedded system does not only receive data. Furthermore, it can also provide data for the student or the student-programmed application which leads to a more interactive and dynamic application. This should motivate students additionally to solve these tasks as proposed in [2]. Therefore, two additional sensors will be connected with the embedded system. The temperature and light sensor values can be evaluated by the students' application which has to determine its further steps depending on the interpretation of the sensor values. Furthermore, students have to calculate a temperature value from the sensor voltage value based on given sensor information.

II. RELATED WORK

In the research of the paper "A Practice-based MOOC for Learning Electronics" [3] the virtual instrument system in reality (VISIR) [4] is used which was developed at the Blekinge Tekniska Högskola (BKH) in Sweden. This approach uses the VISIR system at a MOOC platform. The described MOOC course teaches students in the virtual laboratory electrical basics and allows them to learn by doing without the real hardware and without the risk of damaging the electrical environment by building a short circuit. The results show that the students were a bit worried and skeptical about this learning approach. Nevertheless, after the course, most students thought that this is one of the best ways to teach electrical basics. This is one main reason of this paper to use real hardware to teach students programming languages in a virtual laboratory.

Another virtual remote lab approach is Tele-Lab [5], which offers students the possibility to learn more about the security issue Trojans. Therefore, students are able to attack a virtual machine, which is accessible remotely by a VNC client. The benefit for the students is to see what a Trojan can do on an attacked device from the hacker's perspective. Furthermore, it also sensitizes students how easy it can be to attack another

computer and what should be done to secure computers against these kinds of attacks. It has to be emphasized that this is a practical approach for students to test this security features remotely. Nevertheless, it is not possible to check the students' results and evaluate the students automatically.

A problem in MOOCs and e-learning environments in general is handling large numbers of users. The system has to scale according to the number of active users to prevent bottlenecks and the unavailability of the system especially in virtual laboratories. One approach is a Cloud E-learning and Benchmarking Platform for the Parallel and Distributed Computing Courses [6]. In this paper, this problem is emphasized with the limited availability of cores for each student especially shortly before a task deadline on the server running the user implementations. This problem is also discussed in the "Usage and Evaluation" Section V. A similar approach with real hardware is to adjust the number of necessary real hardware and balance the requests to multiple hardware systems on server side [7] to enhance availability of the system for several students at the same time.

III. APPROACH

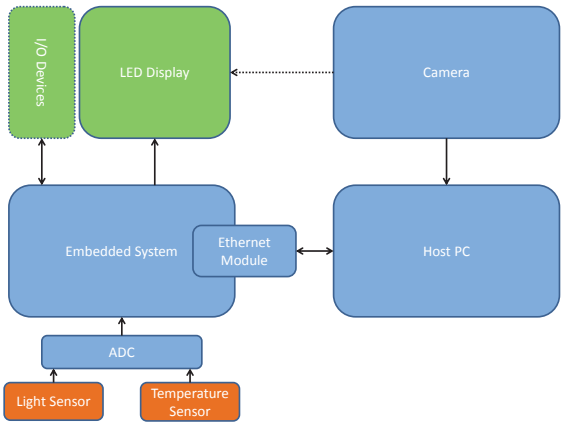

Fig. 1: Overview of the system

A. System Architecture

In this approach, the embedded system uses an additional ADC Module, which allows the connection of sensors to the embedded system. The abstract architecture of this system is visualized in Fig. 1. The embedded system is connected with several devices. These devices are an LED display (16x16 LEDs), an Ethernet module, and a light sensor and temperature sensor which are connected with the embedded system over an analog-to-digital converter (ADC). To enable communication of the embedded system with the MOOC platform a host PC with camera is used. This host PC offers communication with the embedded system over the network. When a student wants to execute his developed application, a request is sent to the host PC. After some reasonability checks the request is sent to the embedded system over the connected Ethernet module. Now the sent message of the student is executed on the embedded system and the result is shown on the LED display. In case of a successful execution of the sent message, the embedded system sends the response "MATRIX OK" to the host PC. Consequently, the host PC is capturing a picture of the LED display and sends the response of the embedded system with the taken picture as response to the request back to the student. In our case, the students use an in-browser programming application called Code Ocean, described in Section C, which is connected by Learning Tools Interoperability (LTI) with our MOOC platform openHPI [8].

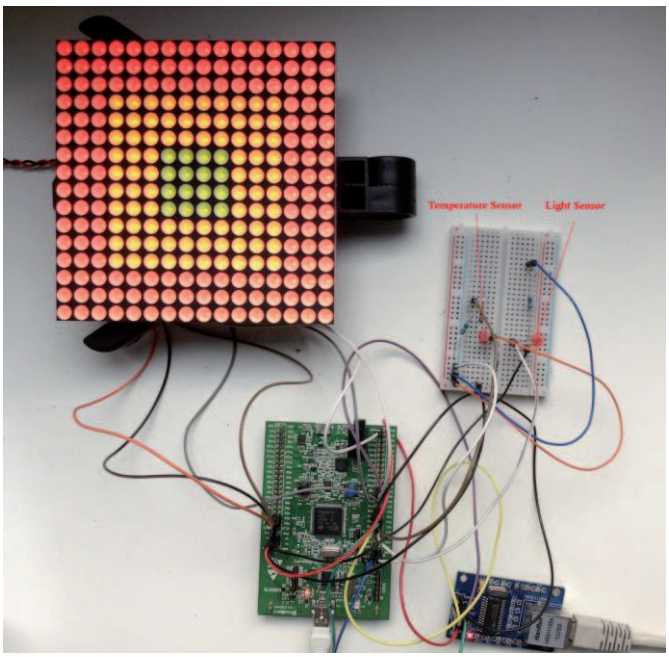

Fig. 2. Setup of the embedded system

B. Implementation

The implemented embedded system of the previously described approach consists of the microcontroller STM32F4DISCOVERY, the LED display SLM1608MD2, the Ethernet module ENC28J60, the light sensor VT93N1 and the temperature sensor LM335Z. Furthermore, there are several cables, a breadboard, a power source and some resistors used to connect the parts properly. The completely connected system with the described hardware is shown in Fig. 2.

1) Board

To implement the STM32F4DISCOVERY board, the programming language C, which is common for embedded systems, is used. During the implementation, the corresponding Input / Output (I/O) pins are used to control the LED display, the Ethernet modules, and the sensors.

First, the LED display has to be handled by the board. Therefore, the following pins of the LED display are connected with the board. These LED display pins are "Red LED", "Green LED", "Clock", "Bright", "Reset", "Select", and "Ground". This pins are being triggered periodically every 10ms by an interrupt of the board. This will ensure that other calculation tasks will be stopped to keep this task running in

time. Otherwise, the display would flicker or turned off completely if another execution task ran and the display would have to wait until the other task is finished. During one interrupt, all 256 LEDs will be set and enabled or disabled depending on the sent array message of the student. Therefore, the "Reset" pin is triggered first to reset the display and guarantee that the display starts to set the LED values from LED 1 and not in the middle. After a reset of the LED display the "Clock" is triggered periodically with every clock signal of the processor. This means the "Clock" pin is changes its status after every clock cycle. During each clock cycle and the corresponding trigger on the "Clock" pin, the LEDs are set from left to right. To set the LEDs the "Red LED" and "Green LED" pins are used for the corresponding color. In case both LED color pins are set, the color will be orange. These colors are set accordingly to the given array by the student. The correlation between the array values and the color is: 0 equals off, 1 equals green, 2 equals red, and 3 equals orange. After each row, the "Bright" pin is going to be set to 1 for a clock cycle to enable (bright) the LEDs with the set values for this row. This is the main reason for the interrupt every 10ms. Otherwise, the LED display would show the student's result only for some milliseconds when a new message is received and would be off the rest of the time. During one interrupt, all rows of the LED display will be set and turned on with the "Bright" pin after each line. In our case, the mentioned "Select" pin is enabled all the time to select the LED display. This is necessary when using more than one display and using the same "Red LED", "Green LED", "Clock", and "Bright" pins for several displays. The "Select" pin decides which display will be set and used currently. Finally, the "Ground" pins of the board and LED display should be connected. Otherwise both devices use a different ground which will lead to unexpected behavior of the system since an enabled and disabled pin cannot be recognized all the time correctly.

In the next step, the implementation for the Ethernet module is described. This module enables the embedded system to communicate with the host PC and finally with the students implementation by the use of a TCP connection. As basis the "ENC28J60 EtherShield Library for Arduino"[1] is used to handle the TCP connection with the board and the ENC28J60 Ethernet module. Therefore, the pin layout has to be adapted in the given library to work with the STM32F4DISCOVERY board. This library handles all incoming TCP connections and checks them for validity. Possible incoming data streams are the following:

- "MATRIX" + 256 digits between 0 and 3 + "X"
- "TEMP"
- "LIGHT"

The incoming "MATRIX" keyword followed by 256 digits and an "X" is used to send a message containing the array values for the LED display. The meaning of every array value between 0 and 3 has already been described in the previous paragraph. The 256 digits describing the expected value for every LED in the LED display matrix. The "X" at the end of the data stream highlights the end of the data and is used to verify the data stream easily. If the verification succeeds, the new array will be drawn on the LED display and the student gets a success message ("MATRIX OK") as answer of the TCP request. In case of an invalid data stream the student gets an error message as TCP response to the request. Other valid TCP message requests are "TEMP" and "LIGHT". These requests will trigger the sensors, which are connected to the board over an ADC and sent as TCP response to the TCP request containing the ADC sensor value. A possible TCP response could be "LIGHT 1121".

To implement the sensor evaluation on this board the ADC module of the board is used. Since this ADC module can only handle voltages up to 2.9V and the temperature sensor can produce values between 0V and 5V we have to find a way to solve this problem. Otherwise, our ADC module will break when voltages over 2.9V will be used as input. As a solution, a voltage divider will be used. This voltage divider consists of two resistors with the same resistance, which is 2.7kΩ in our case. An example of a voltage divider is shown in Fig. 3.

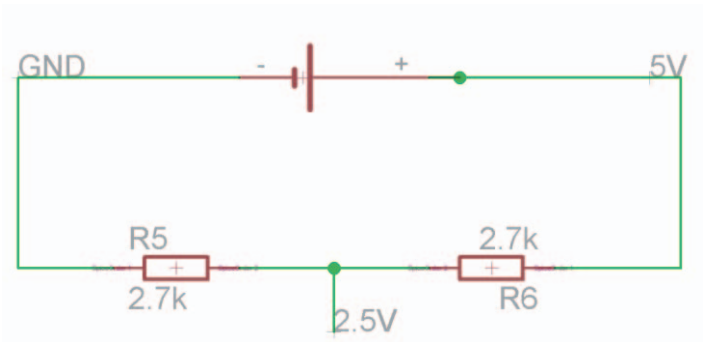

Fig. 3. Example voltage divider

For the light sensor, a voltage divider is not necessary. For this sensor, only one resistor is used like described in the manual for the sensor. Finally, the ADC module is configured to measure the voltage of the sensor and convert it to a digital value between 0 and 4095. This value is calculated into a linear number in the voltage range 0V and 2.9V.

2) Server

The server is the communication center for the coding environment Code Ocean and the embedded system. The host PC checks student requests before they are handled on the final embedded system. Furthermore, the host PC decides under which circumstances an image of the LED display should be captured and when this is not necessary. An invalid request or answer of the embedded system will not trigger an image capturing. Furthermore, also temperature or light requests are not reliable for an image of the LED display. Only

[1] https://github.com/thiseldo/EtherShield

a correct sent matrix triggers this image capture requests. To capture an image fswebcam[2] is used on the Ubuntu 14.04 LTS based server. For the TCP connection a PHP environment[3] is used. This PHP environment is run as a service to accept TCP connections with the help of xinetd[4] small super-server daemon. To handle multiple requests during a MOOC with about 10,000 students the PHP function "flock()" is used to ensure exclusive access to the embedded system and the camera for one student only. Other student requests have to wait to execute the exclusive commands on the embedded system and host PC until the previous student request is finished. Since one student request takes around 2 seconds only, we could not experience serious performance issues on the system.

C. MOOC platform - openHPI

As this experiment took place as part of an online course with a few thousand learners, the dedicated MOOC platform openHPI [8] was utilized to offer the non-interactive assessment modules, namely the instruction text and the instruction video. This platform also hosts a forum, where learners can discuss or search for help while solving the assessments. The MOOC platform was also in charge of tracking the progress of the learners.

The course that hosted the experiment consisted out of six weeks and two bonus weeks. Every week was providing a series of videos, each followed by a multiple-choice selftest. At the end of each week an assessment was provided. At the end of the course a final assessment was offered, covering the topics from all six course weeks. To successfully complete the course and receive a record of achievement at least 50 percent of the 180 total points have to be reached. Every assessment was part of a bonus section of the course and only a small amount of points could be achieved by successfully solving the assessments. Still, the points collected could be used to fill up missing points to achieve a better total result.

While the MOOC platform only provides videos, quizzes, peer assessments and a forum, for this interactive tasks a UI to edit and submit code was needed. The platform's support for integrating external tools based on the LTI 1.1 standard was used to connect to the MOOC platform, which takes care of providing the learner the interface for editing and submitting code. The communication flow of LTI 1.1 is quite simple; the external tool is called with information about the current user and some optional metadata (which exercise should be loaded, is grading still available). Once the user submitted the solution a callback URI is called to store the results in the user's course progress.

[2] http://www.sanslogic.co.uk/fswebcam/
[3] http://php.net
[4] https://github.com/xinetd-org/xinetd

D. Code Ocean

The Code Ocean platform [9] is an open source solution developed at the Hasso Plattner Institute that is able to receive code and execute it in an isolated container environment based on Docker. Based on the container virtualization Code Ocean is able to execute a broad range of coding languages. This is handled by providing several so called execution environments. Each execution environment specifies its container images, hardware restrictions (including max. runtime, CPU and memory limitations, network access) and settings how the code should be called. The execution environment for this experiment was similar to one that was already used, except network access was needed so the code execution container can send the matrix to the experiment host PC. No additional changes were needed to support the use of the sensors.

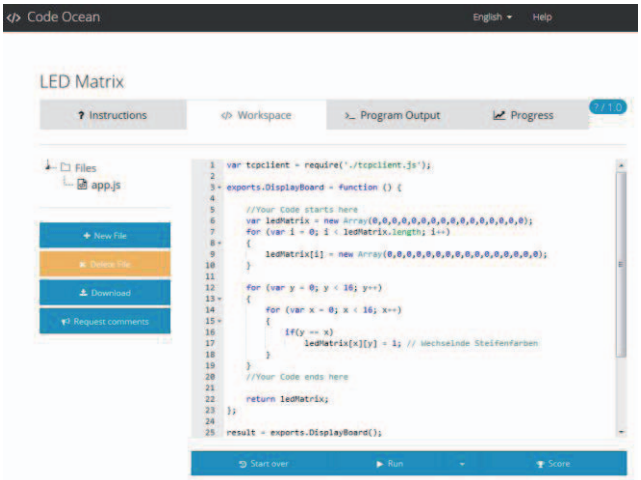

Fig. 4. Code Ocean programming environment, currently running JavaScript

The interface for the learner to enter, edit and submit the code is shown in Fig. 4. It features some additional instructions, so it is not needed to return to the calling LMS or MOOC platform until submissions.

There are two different modes that the learner can execute code. The "run" mode executes the provided code and just sends back and displays the results. Given a simple input code like "print(17+4)" the string 21 would be returned and displayed in the "program output" tab.

For assessments based on the LED Matrix system, the Code Ocean platform was extended, so images could be displayed as part of the result.

While the "Run" command in combination with the display of the picture of the illuminated LED display will allow the learner to receive an instant feedback for grading the assessment a second execution mode called "Score" is provided.

Based on tests provided, as part of the exercise code, the grade is calculated. While the focus on the assessments was not to teach a specific way to achieve the required results, the tests

here could simply check the content of the array that is sent to the LED board. It is notable that for the "Score" mode there is no need to actually use the LED board, as the tests are sufficient to determine if the user has submitted working code.

IV. Experiment

The approach was tested in a Web Technologies course to implement some approaches, which were discussed during the MOOC course in JavaScript using Node.js. During this course, four implementation tasks are prepared to be solved by the students. These tasks are visualized in the Figures 5-8. For every task some basic hidden programming files handling the TCP requests were added to offer a function to send an array to the host PC and the embedded system. Furthermore, unit tests are provided to check students' implementations and use the automatic scoring system.

A. Task 1

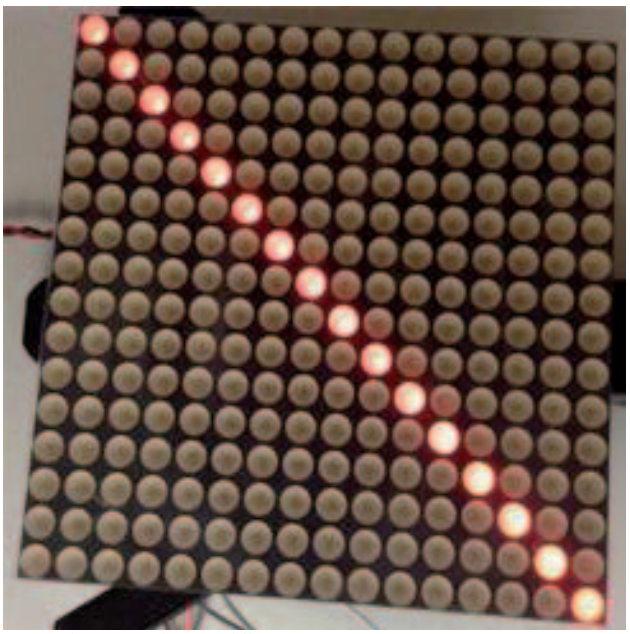

Fig. 5. Task 1 - Drawing a red line

In the first task, the students should understand the basic usage of the embedded system with arrays, and use Code Ocean to implement JavaScript. In this exercise, the students have to draw a red line from left top corner to the right bottom corner. The correct result on the LED matrix is visible in Fig. 5.

B. Task 2

In the second task, three squares should be created by the use of arrays and conditional statements in JavaScript. There are several approaches possible. Students can create it by setting every pixel one by one. Nevertheless, the idea is to find algorithms for arrays to set it correctly with some loops and conditions easily and with less source code necessary. The result looks like shown in Fig. 6.

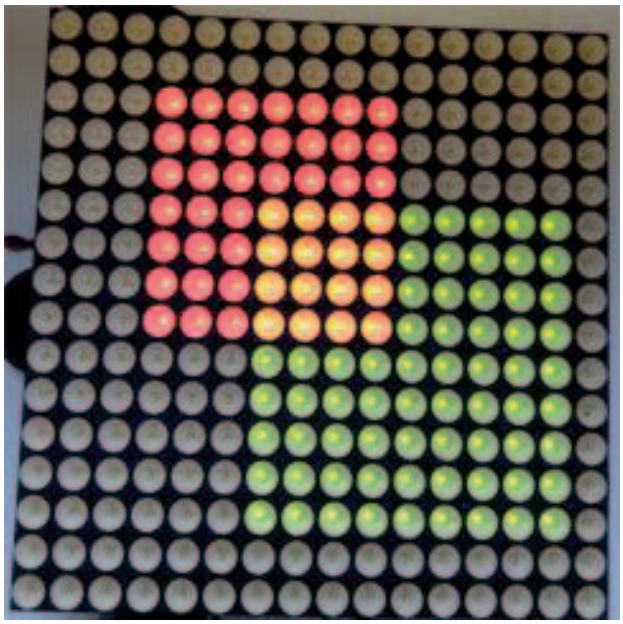

Fig. 6. Task 2 - Drawing several squares

C. Task 3

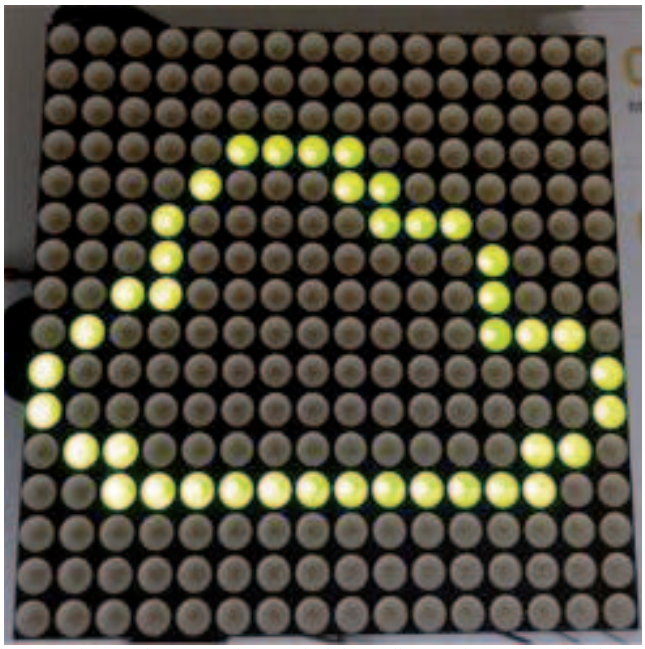

Fig. 7. Task 3 - Select a weather symbol depending on the measured light value

This task starts with the approach to read sensor values and evaluate them. During the third task, the students should learn to read a sensor value from the embedded system and set an already given array for different sensor values. There are different limiting values shown in the following list.

- 3501 - 4095: sun
- 1001 - 3500: sun and cloud
- 301 - 1000: cloud
- 0 - 300: moon

To display the corresponding weather condition the students should use the correct array template according to the given value list. These given array templates contain the possible arrays to show the corresponding weather conditions on the display. An output for the sensor value can be between 0 and 4095. The output for a sensor value 485 is shown in Fig. 7.

D. Task 4

Fig. 8. Task 4 – Draw the measured and calculated temperature value in a given thermometer template on the LED display

The final task should include sensor and an array task. In the last task the students have to use a template and draw the current room temperature given by the temperature sensor. Therefore, a calculation is necessary which describes how the sensor value can be converted to the temperature value in Celsius. The description was: "The ADC value is between 0 and 4095 and represents a voltage between 0 and 2.9V linearly. Every 5mV represent 1 degree in Kelvin. Please round degree Celsius to integers according to mathematical rounding rules."

This way of calculation is typical for embedded systems using analog sensors. Depending on the ADC configuration, the ADC calculates a value representing the voltage based on the actual voltage. According to the voltage, the actual measurement (e.g. temperature) can be calculated depending on the sensor specification. The sensor specification describes the following formula to calculate the temperature. To calculate the temperature, the measured voltage has to be replaced with the "x" and results in

$$temperature\ in\ °C = \frac{x}{0.01V} - 273.15$$

With our ADC configuration the final formula on our embedded system would be the following.

$$temperature\ in\ °C = \frac{2.90V * \frac{x}{4095}}{0.005V} - 273.15$$

Here the "x" describes the digital value of the measured divided voltage after the voltage divider and the ADC conversion. Due to the voltage divider, the steps for 1 degree changed from 10mV per degree to 5mV per degree.

This formula has to be implemented by the students in JavaScript and the result should be drawn with the given template as array template to the LED Matrix. The result for 22°C is shown in Fig. 8.

V. Usage and Evaluation

The approach was tested in a Web Technologies course running on open.hpi.de from June 1st till July, 22nd 2015.

In terms of scalability, no issues occurred during the course. So possible escalation strategies like introducing additional caching or speeding up the processing of the images have not been applied.

Every time a user works on the Code Ocean platform several events are tracked and stored in the database. This includes code runs, code runs against the tests and the final submission to the course platform. As the code ocean platform offers no statistical function yet, the evaluation was based on an export of all code runs. This was also matched with an export of the overall course results to see if any correlations in between the course result and the time effort of the task solution could be identified.

A. Usage

All of the four different experiments out of which the last two featured the handling of input values, were offered. The overall usage of the tasks was slightly lower than those of a different optional module in this course featuring online Ruby coding. The four tasks have been offered in a linear way in the order listed below.

The temperature task, which offered the most real-world approach showed less users finishing the task according to TABLE I.

TABLE I. Details about submissions and earned average points

Task	ID	Users submitted	Avg. Points
Red Line	73	782	0.58 / 1
Squares	74	620	0.70 / 1
Light	75	566	0.43 / 1
Temperature	76	520	0.73 / 2

As shown in Fig. 9 the submission count of the four tasks was very similar for all four tasks. The two peaks corresponded to Mondays. In the MOOC where this tasks have been testes each Monday new content modules have been opened up to the learners. Beside this effect, there was a high usage until the deadline of the submissions.

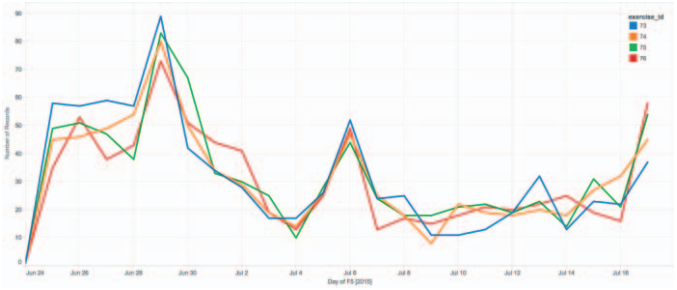

Fig. 9. Submissions per task via the runtime of the course

B. *Tries and Duration*

When looking at the time users needed to complete the tasks, there was a broad range and an extreme long tail, including users who took multiple days from first runs to the final submissions.

TABLE II. Time used from first run to submission

Task	Avg.	Avg. runs/submit
Red Line	34	10.31
Squares	23.4	8.79
Light	34.5	10.52
Temperature	38.8	16.31

Especially the fourth task showed a high average edit time and a high amount of score runs and code runs according to TABLE II.

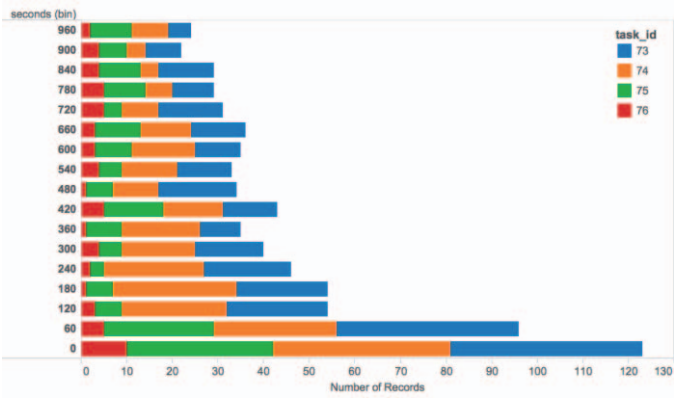

Fig. 10. Time from first code run to last task submissions per task

Not only the average time needed to submit the task was higher, it is also notable that while for the first three tasks many users submitted within a short timeframe, the more complex task four shows only a few of those users. This leads to the conclusion that the task four requires more work like it is visible in Fig. 10.

C. *Impact on course results*

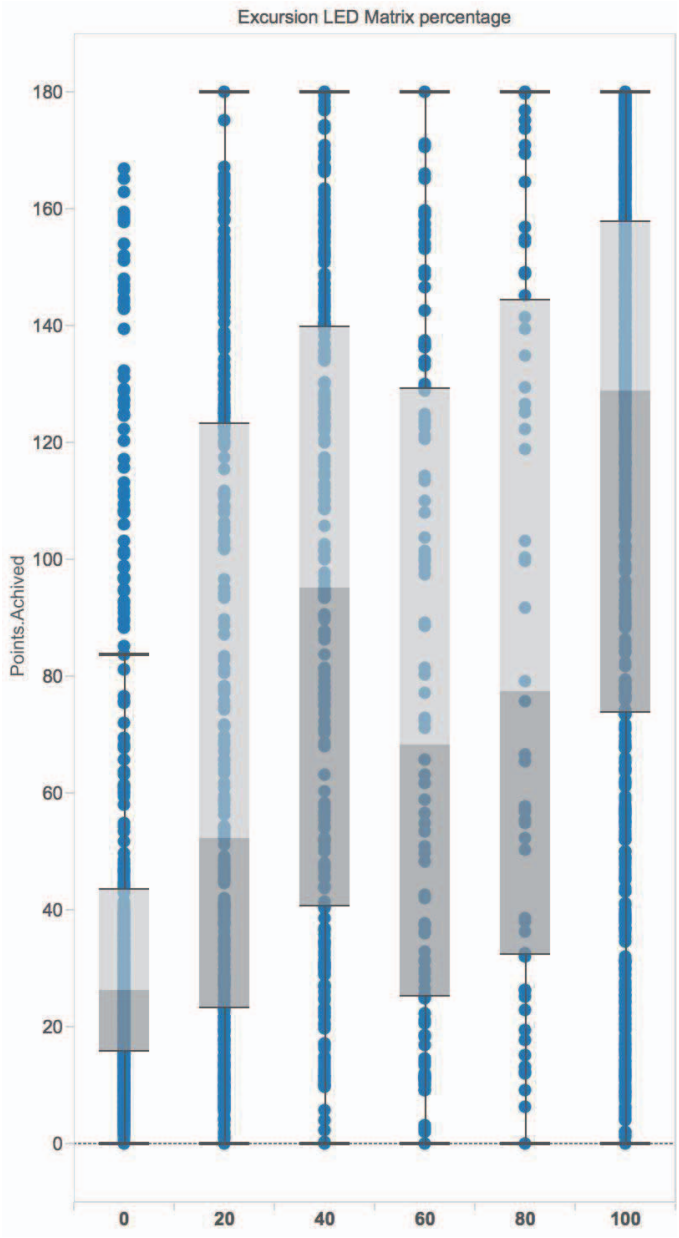

Fig. 11. Visit Progress in the experiment module and overall course progress

Users who visited all items within the experiment section have a better course completion. It is also notable that all users who finished the course with the full amount of points also visited the optional section according to the evaluation in Fig. 11.

If we have a look at Fig. 12, we can detect that users who submitted the most complex task have way better overall results. There is a clear correlation in between their result in this task and their overall course score.

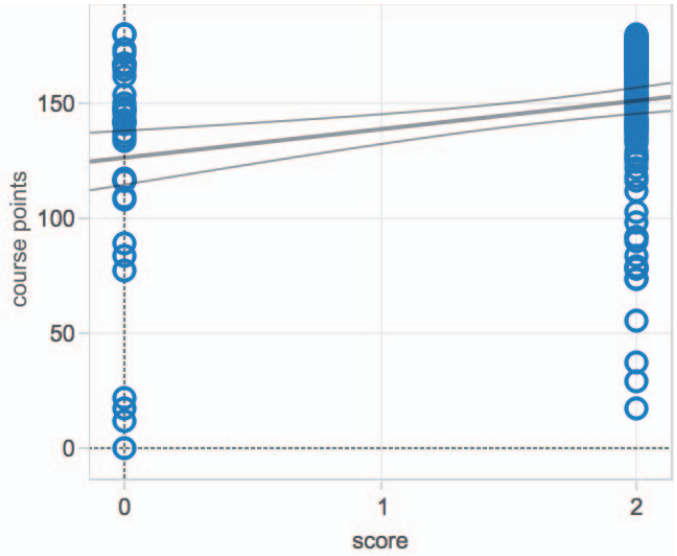

Fig. 12. Correlation of points (temperature task) and overall course score

D. *Qualitative feedback*

According to this analytic feedback the use of this approach was very successful. Furthermore, the students' direct feedback shows the success of this approach. In the discussion forum[5] students evaluate the platform by the following exemplary forum posts.

"Brilliant idea to break out of the virtual world into the physical one. I found the immediate feedback of the picture of the physical LED Matrix highly motivating."

"A really great part of the course. I love that there are not too many hints as this reinforces learning by making you search for answers."

"Excellent exercise, very interesting. If not too complex, would it be possible to publish a set up guide for the LED component? It will be lovely to see the component come to life like magic!"

The last post of the students also shows that we can inspire a student to learn more about using and programming embedded systems since he would like to build the system himself at home.

As only five additional bonus points could be achieved by successfully submitting all four tasks, the motivation to invest the additional time seems to be intrinsic.

VI. Conclusion and Future Work

As a result, this approach shows that the usage of the embedded system increases students' MOOC learning success. Like it was pointed out in Section V the correlation of received points in the additional embedded task and the overall points increase dramatically when students take the time to be able to fulfill the practical exercise.

Additionally, students in this course are more motivated to complete solving tasks on real hardware like emphasized in Section VI D. Some students are also interested in building similar systems on their own since they like the idea of programming real hardware. This leads to future work approaches to offer a MOOC course, which can use real hardware at home and remote accessible hardware.

Providing interactive lab-based exercises can provide a powerful and engaging addition to an otherwise static course. However, we could not prove that the usage of tasks that include input values like temperature or light measurement lead to significant better results.
We could show that using dynamic input values the single exercises could be closer to real world applications.
Based on the user feedback an interesting next step could lead to a hybrid scenario, where some learners would learn on local hardware while others that have no access to certain hardware and no interest in purchasing this hardware could work on a provided lab.

This could be achieved by porting the tasks so they could work on hardware like a Raspberry Pi or similar devices. These devices could be extended by a defined set of extensions such as an LED display or input sensors. To realize this idea the hardware should be easily buyable. In the best case as a hardware package containing all necessary components. Furthermore, the tasks should be realized that it can be executed remotely and on the local hardware in the same way. This makes it possible to check the source code automatically of the students working online and locally. Online working students can check the code and get points by programming in Code Ocean. Locally working students can copy the code after they complete the task into Code Ocean and receive their points the same way.

Finally, an embedded system can improve learning in programming related MOOCs and motivate students in solving additional tasks. This approach shows that it is reasonable to use embedded systems with dynamic input values to motivate and support students learning programming languages remotely.

[5] https://open.hpi.de/courses/webtech2015/pinboard

REFERENCES

[1] Malchow, Martin, et al. "Improved E-learning Experience with Embedded LED System.", to be published

[2] Grünewald, Franka, et al. "openHPI: Soziales und Praktisches Lernen im Kontext eines MOOC." DeLFI. 2013.

[3] Garcia, Francisco, et al. "A practice-based MOOC for learning electronics." Global Engineering Education Conference (EDUCON), 2014 IEEE. IEEE, 2014.

[4] Gustavsson, Ingvar, et al. "On objectives of instructional laboratories, individual assessment, and use of collaborative remote laboratories." Learning Technologies, IEEE Transactions on 2.4 (2009): 263-274.

[5] Willems, Christian, and Christoph Meinel. "Awareness Creation mit Tele-Lab IT-Security: Praktisches Sicherheitstraining im virtuellen Labor am Beispiel Trojanischer Pferde." GI 2008 International Conference on Sicherheit. 2008.

[6] Ristov, Sasko, Marjan Gusev, and Goran Velkoski. "Cloud e-learning and benchmarking platform for the parallel and distributed computing course." Global Engineering Education Conference (EDUCON), 2014 IEEE. IEEE, 2014.

[7] Rasche, Andreas, et al. "Foucault's Pendulum in the Distributed Control Lab." Object-Oriented Real-Time Dependable Systems, 2003. WORDS 2003 Fall. The Ninth IEEE International Workshop on. IEEE, 2003.

[8] Meinel, Christoph, and Christian Willems. openHPI: das MOOC-Angebot des Hasso-Plattner-Instituts. Vol. 79. Universitätsverlag Potsdam, 2013.

[9] Staubitz, Thomas, et al. "CodeOcean - A Versatile Platform for Practical Programming Excercises in Online Environments.", to be published