An attempt to adopt DevOps on embedded system development: empirical evidence

Purnama Eka Wijaya$^{1,2}$, Imron Rosyadi$^1$, Acep Taryana$^1$

$^1$ Electrical Engineering Department, Universitas Jenderal Soedirman, Indonesia

E-mail: $^2$official.purnamaekawijaya@gmail.com

Abstract. DevOps is a method that integrates and narrows two environments in software development process, developer side and operational side. DevOps becomes a solution that can be offered to improve SDLC (System Development Life Cycle) in current embedded system development process. This paper provides a concept and its implementation to prove that DevOps adoption on embedded system architecture is possible. After doing the research, we found out that it is possible to adopt DevOps on embedded system development, we also provide empirical evidence and analyze the weaknesses about the concept designed.

1. Introduction
Embedded system works for specific purposes, this system usually contain software that embed on its controller to set its hardware function electrically [1]. Embedded system applications easily found in everyday products and industrial equipment in many aspects.

In the era of industrial revolution 4.0, internet access is affecting system development process including embedded system [2]. Embedded systems must be able to keep up with current technological developments that occur every second. Embedded system development can no longer use classic SDLC (System Development Live Cycle). Function update delivery, release management, integration, data exchange, and understanding between developers are weaknesses that must be handled by upcoming embedded system SDLC [3].

DevOps (Development and Operation) is a software development and application delivery method that narrows distance between developer and operational side [4]. The idea of DevOps is to build or develop applications that are more integrative, have a low failure rate, and correct errors faster. DevOps becomes possible to be applied in embedded system development process because embedded systems software quality directly determine its hardware functionality.

DevOps becomes a solution that can be offered to overcome the SDLC problem on today's embedded systems development. But the scarcity of platform that can handle the delivery of features to operational environment and monitor the system performance have made empirical evidence of DevOps implementation in embedded systems domain as far as we know are not found yet [5].

To prove DevOps adoption on embedded systems empirically, we try to design an embedded system architecture concept which in its development adopts DevOps principle. We hope that this research will revolutionize today’s embedded system development technology.
Contribution that can be given by this paper are first, presenting a concept of DevOps adoption on embedded systems domain; second, provide empirical evidence of DevOps adoption on embedded systems development; third, specifically identify the weaknesses of the concept designed by the authors.

2. Background

2.1. The Concept of DevOps
DevOps integrates two environments in software development process, developer side and operational side [4]. DevOps is very closely related to agile, DevOps even adopts agile development principle but in a wider scope [6]. The most obvious difference is that DevOps not only enhances collaboration between developers as in agile, but also involves operation team [7]. DevOps can also provide an ideal and efficient circumstances for end users and companies by implementing lean philosophy which maximizes productivity and reduce resources waste [8]. DevOps implementation can also shorten the feedback loop on SDLC through collaboration and automation [9].

With DevOps, Dev (developer side) and Ops (operational side) section collaborates in cross-functional and simultaneous ways. This concept provides advantages for both sides because it provides continuous updates, resolves errors faster and reduces miscommunication that occur within team [4]. DevOps revolutionized many things including service of product into Software as a Service (SaaS), change collaboration culture to be more integrated, and more sustainable application deployment process [10].

Automation holds an important role in creating efficient collaboration and integration between Dev and Ops [11]. By testing, deploying, operating, and monitoring an application automatically, DevOps make application development more targeted and effective according to market demands [4].

DevOps affects team structure and coordination in an industry [12]. Software and IT industry is concerned with speed and efficiency. DevOps has emerged as a paradigm for delivering innovative products and features faster to market. Many related technologies have recently emerged to facilitate in to DevOps because of its fluid delivery practices that can manage complexity [4].

In past, someone had to buy a complete bundle of software to get major update, but now it’s different because developers can build or repair a value faster and continuously send it to the end users in a minor update so that changes can be immediately seen [13].

We find that there are 5 important concepts of DevOps:
1. Continuous Development, Planning is often done to overcome rapid changes in the business environment [14]. Development of a blueprint, define product development vision, and continuous software delivery based on feedback and errors is the key of continuous development.
2. Continuous Testing, allows direct testing when there are changes on updated source code. Testing is done to find out and handle errors as quick as possible [15].
3. Continuous Integration, enables integration at each stage of software development. This concept holds an important role because if there is an error then the changes will not be forwarded to further DevOps stage [16].
4. Continuous Deployment, enables automatic software deployment when there are new updates in a certain time to end users [17]. It increases productivity and reduce errors delivery risk [18].
5. Continuous Monitoring, make it possible to monitor application being operated to analyze system performance. Collected data can serve as feedback for developers [19].

Platform used to ensure all concepts are met is different and specific for each concept. The platform also needs to be integrated and has plug-ins that can be accepted with each other.

DevOps has been largely adopted by cloud-based applications companies such as Facebook, Google, and Amazon [9], [13]. DevOps has proven successful in implementing SaaS, because the company maintains full control and has quick mechanism to relaunch new software releases whenever there is problem [5].
2.2. DevOps On Embedded System
Most DevOps studies focus on web domain in many architectures and scenarios. DevOps adoption on embedded system poses more challenges because of different architectures and automation technologies that are mostly based on cloud platforms and informatics domain [4]. Furthermore, regulated tools that can be used as a standard for specific uses is needed [20], [7]. We have never seen research that really provides empirical evidence in case of DevOps adoption on embedded systems [5].

There are many factors that influence DevOps adoption on embedded systems, such as development costs, development plan made by previous developers, and system reliability [21]. Companies or developers need to determine technology transfer that affects system development cultures. Moreover, the scarcity of platform that specifically handle feature deployment to the operational environment and automatically monitor system performance also presents significant challenges [5].

Trend shows that embedded system product development is proven to be in hardware aspect [21], while software development can also maximize hardware functionality. Separation of hardware and software development also extends development cycle time [5]. Large amount of time must be spent in architectural design to describe complete system functionality rather than focusing solely on urgent needs of software update [21].

3. Method

3.1. Adoption Concept
Classic architecture of embedded system consist 3 main parts: input, controller, and output. To prove that embedded system development can be done with DevOps, The adoption concept not only must be able to improve its software quality but also its hardware functionality. Changes in the performance of sensors and actuators can be a proof that embedded system development using DevOps have been adopted.

![Figure 1. Adoption concept](image)

Marriage between embedded systems and DevOps occurs in operations section when application is deployed and operated on an embedded system controller, the concept is visualized by Figure 1. The application deployed to the embedded system controller must be able to access all forms of hardware integrated in the controller both input and output.

DevOps adoption concept on embedded system have two requirements. First, embedded system controller should have an operating system so that it would be easier for container to work on its kernel. Second, DevOps adoption should be developed by right platform.
VCS (Version Control System) is a platform that can support collaborative software development. It makes software development being parallel, so that several developers can work on different function codes for the same application. VCS works to accommodate the concept of continuous development.

Automation server is a platform that can be used to automate all kinds of tasks related to building, testing, and integrating software. Automation server works to accommodate the concept of continuous testing and continuous integration.

Container is a platform that can be used to provide isolated systems (isolated environments) at OS level which run on parent kernel. This platform allows developer to package all application needs in a container and run it on various kernels. Container contents can be changed as update goes on so that it allows sustainable application deployment to destination address, in this case an embedded system controller. Container works to accommodate the concept of continuous deployment.

Monitoring server works to carry out data collection process on a system and conducts an analysis of these data to maximize all available resources and giving feedback for developers. Server monitoring works to accommodate the concept of continuous monitoring.

Platform used by the authors can handle all stages of DevOps method as shown in Figure 2. Such configuration is a common thing used in developing web-based applications in DevOps method.

### 3.2. Physical Development

To determine platform compatibility on embedded systems, we conducted a small experiment to compare the platforms used, it shows in Table 1 below.

| Platform                      | Web based software Development | Embedded system Development |
|-------------------------------|--------------------------------|-----------------------------|
| VCS                           | Work properly                  | Work properly               |
| Automation server             | Work properly                  | Work properly               |
| Container                     | Work properly                  | • Need to install container on controller OS  
|                               |                                | • Need to set SOC core type specifically  
|                               |                                | • Need to use correct hardware library  |
| Monitoring server             | Work properly                  | • Built by Ops because there is no specific tool to monitor embedded system performance  
|                               |                                | • Need to bring physical output data to digital form  |

![Figure 2. Platform used to accommodate the concept designed](image-url)
In physical development shows by Figure 3, Several platforms such as GitLab, Jenkins, and Docker are not mandatory, there are several other platforms that can handle the process which has similar capability. Web hosting is used as monitoring server prototype because there is no monitoring server that specifically handles embedded system monitoring stage on DevOps. Those platforms are used because they are open source and commonly used in DevOps web-based software development process.

The process starts when the latest code development is finished by Dev. After the code is finished it will be pushed to GitLab repository VCS, then automated build and test will be carried out by Jenkins automation server. If there is no structural error in the code, Jenkins will release the code to container

**Figure 3. Physical Development**
hub in this case Docker hub. After the code has arrived at Docker hub, deployment process will be generated by Docker client on embedded system controller by pulling the code and operate it. Before Docker client operates the code, it will be re-tested, if it’s compatible then the code will be run.

In this research we produce two embedded systems. The first one is a simple system which serves to turn on LED (Light Emitting Diode) when push button is pressed and the second product is a system that automatically calculate human BMI (Body Mass Index).

In the first embedded system, we will update the code to manipulate the number of flashing LED when push button is pressed. LDR (Light Dependant Resistor) functions as a dedicated device that will respond to number of flashes. In the second embedded system, we will update the code to manipulate function that responsible for height and weight measurement. Generated data will be sent to monitoring server as a record on database. Ops will analyze the data and report it as feedback to Dev by adding an issue or “to do” list on VCS to be responded in further system development.

3.3. Performance Analysis

3.3.1. Technology Demonstrator

Based on the physical development described earlier, we will demonstrate the changes that occur in two systems that we produced. Figure 4 and Figure 5 are screenshots of a monitoring server that display the data sent by those two embedded systems. Figure 4 shows that there are 3 data, the first one shows that the LDR is sensing 1 signal from the LED, the second one is sensing 2 signal, and the third one is sensing 3 signal from the LED. The data represents 3 application updates that are operated on the first embedded system through the concept designed by the authors. Figure 5 shows that there are 4 data produced by the same person. It can be seen that the second and third data shows the person’s weight and height (BB and TB) that are not relevant compared to the first and fourth data because the authors is manipulating the function formula. The fourth data is the data generated by the fourth deployment to reverse it to the correct function.

![Figure 4. First embedded system monitoring server](https://www.monitoring.firmandev.tech)

| NO | WAKTU         | RFID | TB  | BB | BMI | AB | BLA | BIB | TBA | TBB | KBA | KBB |
|----|---------------|------|-----|----|-----|----|-----|-----|-----|-----|-----|-----|
| 1  | 2019-04-09 11:31:19 | 2506######## | 176 | 76 | 24.54 | 3 | 77.44 | 57.31 | 0 | 0 | 0 | 0 |
| 2  | 2019-04-09 11:45:37 | 2506######## | 174 | 5 | 1.65 | 1 | 75.69 | 56.01 | 70.69 | 51.01 | 0 | 0 |
| 3  | 2019-04-10 14:48:17 | 2506######## | 170 | 91 | 31.49 | 5 | 72.25 | 53.46 | 0 | 0 | 37.54 | 18.75 |
| 4  | 2019-04-10 15:40:00 | 2506######## | 176 | 76 | 24.54 | 3 | 77.44 | 57.31 | 0 | 0 | 0 | 0 |

![Figure 5. Second embedded system monitoring server](https://idealis.firmandev.tech)

![Figure 6. Container status](https://root@raspberryypi/home/pi# docker ps)

| CONTAINER ID  | IMAGE               | COMMAND                                      | CREATED           | STATUS         |
|---------------|---------------------|----------------------------------------------|-------------------|----------------|
| 7672c3807b9a  | pyuroboros/pyuroboros | "ouroboros --interval /bin/bash ./PROGRAM..." | 12 seconds ago    | Up 10 seconds  |
| 3b0d62f771cf  | pyuroboros/pyuroboros | "ouroboros --interval /bin/bash ./PROGRAM..." | 2 minutes ago     | Up 2 minutes   |
Figure 6 is a container status which can be seen on embedded system controller terminal, it shows that the latest code was successfully deployed and has been operated on embedded system controller.

3.3.2. Adoption Concept Implementation
The concept implementation of DevOps adoption on embedded systems answers to technical and managerial problems that occur in the development process, it shows in Table 2 below.

| Implementation |
|----------------|
| **Stages**     |
| Most of the stages (Plan, code, build, test, release, deploy, operate) can be done using web-based DevOps platform such as VCS, automation servers, containers |
| Test stages can only accommodate structural errors while embedded systems testing scenarios are specific and varied |
| Monitor stage does not have specific platform to accommodate embedded system output data collection |
| **Teamwork, collaboration, and interaction** |
| Dev and Ops have their own portions in application development stages |
| Dev and Ops collaborates in cross-functional and simultaneous ways |
| Application created by Dev will be responded by Ops feedback |
| **Deployment** |
| Container deployment can be done accurately |
| The latest application can be deployed to more than one embedded system |
| The latest application can be deployed as soon as the latest code reaches container hub |
| **System performance monitoring** |
| Ops made a monitoring server prototype according to system needs |
| Output data from embedded systems have to be sensed by dedicated device and then brought it to the digital form to be analyzed |
| Ops provide feedback in form of issue or “to do” list to Dev |

4. Conclusion
In this research we provide a concept and practice of implementing DevOps on embedded systems to prove that DevOps adoption on embedded system architecture is possible. First, most literature studies in our research are concepts, reports, and analysis. There is no technical proof of DevOps adoption on embedded system. We find that it is very possible to adopt DevOps on embedded system development, especially in deploy and operate stage. The adoption concept designed is successfully applied since the latest applications has been automatically deployed and operated on embedded system controller. DevOps implementation also provides many enhancements in various variables of embedded system development.

Second, Embedded system produced by the authors directly provide empirical evidence of the adoption concept designed. This is evidenced by the success of application updates that controlling embedded system input and output peripherals.

Third, there are some weaknesses that are technically had to be examined in further study. Compatibility of sensor libraries used causes frequent failures when deployed applications to be operated on controller. Embedded system performance monitoring still depends on system built by Ops because of its specific purposes. Testing mechanism is also a problem, platform used by the authors can only test structural error while embedded system testing scenario is very diverse and specific.
References

[1] Michael Barr and Anthony Massa, Programming Embedded Systems: With C and GNU Development Tools, 2nd ed, O'Reilly Media, Inc, 2006.

[2] F Almada-Lobo, "The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES)" Journal of Innovation Management, vol. 4, no. 3, pp. 16-21, 2015.

[3] P K Ragunath, S Velmourougan, P Duvachelvan, S Kayalvizhi, and R Ravimohan, "Evolving A New Model (SDLC Model-2010) For Software," International Journal of Computer Science and Network Security, vol. 10, no. 1, pp. 112-119, 2010.

[4] Christof Ebert, Gorka Gallardo, Josune Hernantes, Nicolas Serrano, "DevOps," May 2016. [Online]. Available: https://www.computer.org/csdl/magazine/so/2016/03/mso2016030094/13rRUNvyary. [Accessed 1 July 2019].

[5] Lucy Ellen Lwakatare, Teemu Karvonen, Tanja Sauvola, Pasi Kuvaja, Helena Holmström Olsson, Jan Bosch, Markku Oivo, "Towards DevOps in the Embedded Systems Domain: Why is It so Hard?," Hawaii International Conference on System Sciences, vol. 49, pp. 5437-5446, 2016.

[6] Amol S Patwardhan, Jon Kidd, Tiffany Urena, Aishwarya Rajgopalan, "Embracing Agile methodology during DevOps Developer Internship Program".

[7] Anand Srivatsav Amaradri, Swetha Bindu Nutralapati, Continuous Integration, Deployment and, Karlskrona: Blekinge Institute of Technology, 2016.

[8] Ahmed Bahaa Farid, Yehia Mostafa Helmy, Mahmoud Mohamed Bahloul, "Enhancing Lean Software Development by using," International Journal of Advanced Computer Science and Applications, vol. 8, no. 7, pp. 268 -277, 2017.

[9] D G Feitelson, E Frachtenberg, and K L Beck, "Development and Deployment at Facebook," IEEE Internet Computing, vol. 17, no. 4, pp. 8 -17, 2013.

[10] Floris Erich, Chintan Amrit, Maya Daneva, "Report: DevOps Literature Review," ResearchGate, 2016.

[11] Johannes Wettinger, Uwe Breitenbuecher, Frank Leymann, "Standards-based DevOps Automation and Integration Using TOSCA," in International Conference on Utility and Cloud Computing, Stuttgart, 2014.

[12] L E Lwakatare, P Kuvaja, M Oivo, "Dimension of DevOps," 16th International Conference on Agile Software Development, pp. 212 -217, 2015.

[13] Liming Zhu, Len Bass , George Champlin-Scharff, "DevOps and Its Practices," IEEE, 25 April 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7458765. [Accessed 2 July 2019].

[14] Johanna Bragge, Hilkka Merisalo-Rantanen, Petri Hallikainen, "Gathering Innovative End-User Feedback for Continuous Development of Information Systems: A Repeatable and Transferable E-Collaboration Process," IEEE TRANSACTIONS ON PROFESSIONAL COMMUNICATION, vol. 48, no. 1, pp. 55 - 67, 2005.

[15] David Saff, Michael D. Ernst, "Reducing wasted development time via continuous testing," in International Symposium on Software Reliability Engineering, Cambridge, 2009.

[16] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, Juhani Warsta, Agile Software Development Methods: Review and Analysis, Espoo: VTT publication, 2002.

[17] Helena Holmström Olsson, Hiva Alahyari, Jan Bosch, "Climbing the “Stairway to Heaven” A multiple-case study exploring barriers in the transition from agile development towards continuous," Euromicron Conference on Software Engineering and Advanced Applications, vol. 38, pp. 392 - 399, 2012.

[18] Tony Savor, Mitchell Douglas, Michael Gentili, Laurie Williams, Kent Beck, Michael Stumm, "Continuous Deployment at Facebook and OANDA," IEEE International Conference on Software Engineering Companion, vol. 38, pp. 21-30, 2016.
[19] J F Perez, W Wang, and G Casale, "Towards a DevOps Approach for Software Quality Engineering," in Workshop on Challenges in Performance Methods for Software Development, pp. 5 - 10, 2015.

[20] Teemu Laukkarinen, Kati Kuusinen, Tommi Mikkonen, "DevOps in Regulated Software Development: Case Medical Devices," in Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Results Track, vol. 39, no. 20, pp. 15 - 18, 2017.

[21] B Graaf, M Lormans, and H Toetenel, "Embedded software engineering: The state of the practice," IEEE Software, vol. 20, no. 6, pp. 61 - 69, 2003.