**Deliverables**

|  |  |  |
| --- | --- | --- |
| **Activity name:** Minutes of Meeting | | **Delivery date:** Weekly |
| **Input:** Previous Minutes of Meeting, Agenda | | |
| **#1** | **Activities:** | |
| For every meeting the team has a Minutes of Meeting shall be published. The document shall describe how the meeting went, what topics were discussed and what decisions were made, as well as who participated in the meeting, where it took place and at what time. The minutes of meeting shall be updated after every meeting so that members who could not participate in the meeting for one reason or another have the opportunity to catch up. | | |
| **Output:** This week’s Minutes of Meeting | | |

|  |  |  |
| --- | --- | --- |
| **Activity name:** Project plan | | **Delivery date:** 21.02.2020 |
| **Input:** Use cases, deliverables, risk analysis, project structure and organization | | |
| **#2** | **Activities:** | |
| The project plan includes everything regarding the work of the team, including how the team will function, how the project will be organized, what the team is expected to deliver, how communication with the client takes place and an analysis of the risks. | | |
| **Output:** A general plan of how the project will proceed | | |

|  |  |  |
| --- | --- | --- |
| **Activity name:** Sensor calibration | | **Delivery date:** N/A |
| **Input:** Sensors provided by customer | | |
| **#3** | **Activities:** | |
| After the humidity, temperature and particulate matter sensors are provided by the customer the team shall set up the mentioned sensors for testing and calibration. | | |
| **Output:** Working sensors | | |

|  |  |  |
| --- | --- | --- |
| **Activity name:** Building the network | | **Delivery date:** N/A |
| **Input:** ZigBee module | | |
| **#4** | **Activities:** | |
| The team shall use the ZigBee module provided by the customer combined with an embedded board to build a simple network used for communication between the sensors and the C# application. | | |
| **Output:** Functioning network | | |

|  |  |  |
| --- | --- | --- |
| **Activity name:** C# application | | **Delivery date:** N/A |
| **Input:** Functioning network, sensors | | |
| **#5** | **Activities:** | |
| A C# application will be built to simulate a ventilation box. The application will take the readings from the sensors over the network and communicate with the sensors bi-directionally. Additionally, the final assembly of the system will take place. | | |
| **Output:** Working application | | |

**Risk analysis**

In this project we have defined the following risks:

1. Bad time management. Even though everything is scheduled and always updated in git, the possibility that someone forgets to check their scheduled times for meeting is still there.
2. Not meeting due dates for deliverables. Programming and embedded systems are not always straightforward and things almost never go according to plan. Additionally, it could be the case that someone is having troubles with their assignment, which could further slow down the process.
3. Sprint demo failing. Sprint demos are always botched together solution just for demonstration purposes and prototyping. Thus there always exists the possibility that the demo fails, which will leave the client dissatisfied.
4. A project team member fails to deliver their part. Perhaps someone will have struggles with their task, maybe personal problems or lack of organization, anything can go wrong even on a personal level. This could lead to significant delays.
5. Documentation is incomplete. The team members are already quite familiar with the documentation process, however there could still be some holes in the process. Mentor’s input should be considered at all times.
6. Provided hardware malfunctions. The solutions the team is going to be working with are well-tested and stable products, however hardware is hardware and something could always go wrong.
7. The needed hardware is not provided on time by customer. This is always a possibility when the team relies on the customer to provide something.

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Probability** | **Impact** | **Mitigation** |
| 1. Bad time management | Medium | Medium | Every meeting is scheduled in advance and everyone’s preferences are taken into account |
| 1. Not meeting due dates for deliverables | Low | High | Due dates are carefully tracked and scrum master makes sure everything is delivered on time |
| 1. Sprint demo failing | Medium | High | Sprint demo prototypes are carefully tested a day before the event |
| 1. A project team member fails to deliver their part | High | Medium | Scrum master keeps track of how everyone’s handling their tasks and offers help if needed. |
| 1. Documentation is incomplete | Low | Medium | We make sure to keep the client and the mentor updated to make sure everything is delivered correctly. |
| 1. Provided hardware malfunctions | Low | High | In case of any issues the client is to be contacted and new hardware is requested as soon as possible. |
| 1. The needed hardware is not provided on time by customer | Low | High | In case of this event the team is responsible for supplying needed hardware to finish on time. |

**Configuration management**

As established, all of the code written by the team members should be submitted to the git repository. From there, a member of the team should review the code and give feedback. The program should then be tested and possibly a new iteration of the program will emerge, which also has to be reviewed, tested etc. If a member wants their code to be reviewed and tested, they can assign a task in “Boards” in gitlab or simply write a message in the WhatsApp group. When an application is assigned to more than 1 person, the members are responsible for creating their own branch to work on in git instead of using the master branch. After each of their codes are finished, reviewed and tested, they are both responsible for merging their branches to the master branch, which contains a stable version of the project, which is suitable for sprint demos.