TRƯỜNG ĐẠI HỌC FPT

CAPSTONE PROJECT DOCUMENT

AMAZING BIKE

GROUP VII
Nguyễn Trọng Thịnh - SE03170
Vũ Tiến Trung – SE03102
Trần Văn Tuấn – SE03152
Vũ Quang Quyền – SE3237
Lưu Ngọc Việt Sơn - SE03711

JULY 4, 2016

ACKNOWLEDGEMENTS

We couldn’t have finished this project without the help and technical assistance of many people. In particular, we would like to thank our family. The amount of time and effort that goes into this project is hard to measure.

Especially, we truly had a fantastic person helping us - Mr. Hoang Xuan Son, his professional guidance and his experience help us to overcome the obstacles, even the hardest time when we think we can’t continue with this project anymore. He has truly had a significant impact on the success of this project.

We would like to extend a special thank you to our teacher, friends who have support us throughout 4 months of this project.

Table of Contents

I. Introduction:

1. Project information:

- Project name: Amazing Bike
- Project code: AB
- Project type: Embedded System and Android application
- Start date: March 22th 2016
- End date: August 31th 2016

2. Project member:

No	Name	Role	Contact
1	Hoàng Xuân Sơn	Supervisor	Phone: Email: SonHX@fpt.edu.vn
2	Nguyễn Trọng Thịnh	Leader	Phone: 0167 2559 886 Email: ThinhNTSE03170@fpt.edu.vn
3	Vũ Tiến Trung	Member	Phone: 093 4217 251 Email: TrungVTSE03102@fpt.edu.vn
4	Trần Văn Tuấn	Member	Phone: 097 3322 413 Email: TuanTVSE03152@fpt.edu.vn
5	Vũ Quang Quyền	Member	Phone: 0167 4092 091 Email: QuyenVQSE03237@fpt.edu.vn
6	Lưu Ngọc Việt Sơn	Member	Phone: 091 2872 336 Email: SonLNVSE03711@fpt.edu.vn

Table 1- Team member

3. Problem:

Today, the progress of modern science has an impact on all aspects of human life. And in which the growth of traffic, especially the transport has flourished to help people be able to move freely and safely. We already have electric cars that are environmentally friendly, self-driving cars of Tesla help people in traffic safely, comfortably.

Figure 1: Electric car

Figure 2: Self-drive car of Tesla

Currently, people are very interested in the transportation of human friendly. In addition to that the self-propelled mechanism it can also help people in traffic safely while they can still do other work.

Figure 3: Self-drive car

Besides all kinds of modern transport, the bicycle as a means of transport is indispensable in today's life. It just helps to avoid traffic jams in big cities, but also a means to help people improve health modernization.

Figure 4: Bike cycle in the city

4. Existing Solution:

4.1 Smart Balance Wheel:

As personal transportation, suitable to move in large spaces, walking neighborhoods or urban area, can go shopping in large supermarkets, or delivered in the large lobby.

Figure 5 - Smart balance wheel

Disadvantages: for people with disabilities, difficulty in standing, or not to keep the balance, or those who prefer sports activities, then this is not a good choice for them.

4.2 Ninebot – Personal Transportation Robot:

Figure 6 - Ninebot – Personal Transportation Robot

Ninebot is the new smart moving vehicle of Segway- one brand of US. It has the ability to move faster than walking 4 times, easy to move on the road with 15 degrees slope and rocky road, can go a long way 30km.

It helps you move quite convenient but it's quite voluminous and those who prefer to movement, this is not the first choice.

5. Ideas:

With a bicycle self-balanced, self-moving, people with disabilities can participate in traffic with ease, even with healthy people need exercise, the car is still not a bad choice. Moreover, subsequent to the development of self-propelled mechanism, the vehicle can move to the location was chosen before, very suitable for the delivery of automatic cave without shipper, as well as shuttle children to kindergarten or carrying them home safely.

Figure 7 - Self-balance bicycle

Therefore, in order to express the passion of science, as well as creative inspiration, favorite robot programming department in particular and in general Embed FPT University students. And particularly complete learning program at FPT University, we have decided to implement the scheme self-balancing bike to complete the above purposes.

6. Feature Function:

The system has two main parts: self-balancing bike moving and self-balancing bike while moving with the control of the smart phone running Android.

- Bicycles balance when moving itself.
- Bicycles can turn left, turn right with the control of smartphone.

II. System Project Management Plan (SPMP):

1. Purpose:

The purpose of this chapter is to describe the organization and plan of the project. All team members must use this chapter as a guideline for tracking assigned tasks and deadlines. This chapter also included an overview of this project and team member. This is a document for daily meeting and meeting minute.

2. Project Organization:

2.1 System Process Model:

Our project uses the Iterative and Incremental Software Process Model.

Iterative and incremental software development is a method of software development that is modeled around a gradual increase in feature additions and a cyclical release and upgrade pattern.

Iterative and incremental software development begins with planning and continues through iterative development cycles involving continuous user feedback and the incremental addition of features concluding with the deployment of completed software at the end of each cycle.

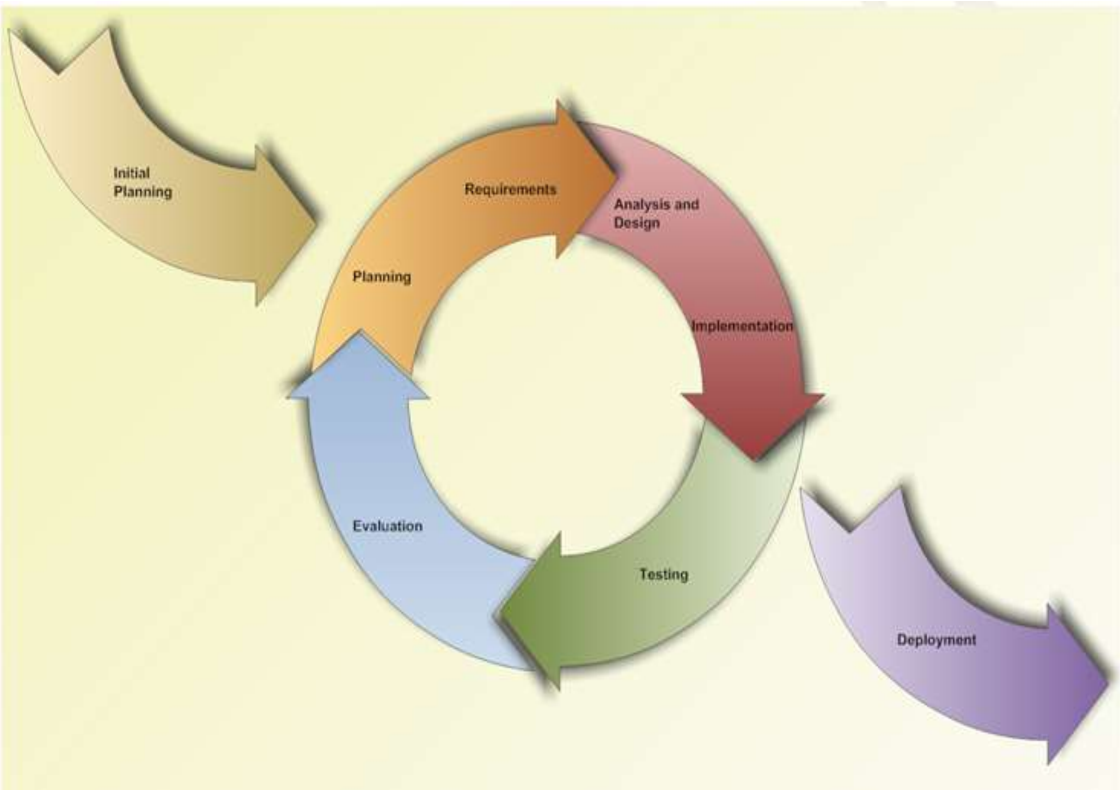


Figure 8 - Iterative and Incremental Software Process Model

2.2 Roles and Responsibilities:

No	Name	Role	Responsibilities
1	Hoàng Xuân Sơn	Supervisor	- Approving and supporting process to run project. - Suggesting solution when the project meets issue.
2	Nguyễn Trọng Thịnh	Project Manager	- Select methodology to identify risk. - Set common rule to all members in project to avoid risk. - Approve solution to resolve issues.
3	Vũ Tiến Trung	Technical Leader	- Investigate technical issues. - Review source codes.
4	Trần Văn Tuấn	QA Leader	- Keeping all of member on process and follow common rule. - Reviewing quality of resolved issues.
5	Vũ Quang Quyền	Developer	- Follow process of project. - Follow common rule of project. - Raise issue to leader in team meeting.
6	Lưu Ngọc Việt Sơn		

Table 2- Roles and Responsibilities

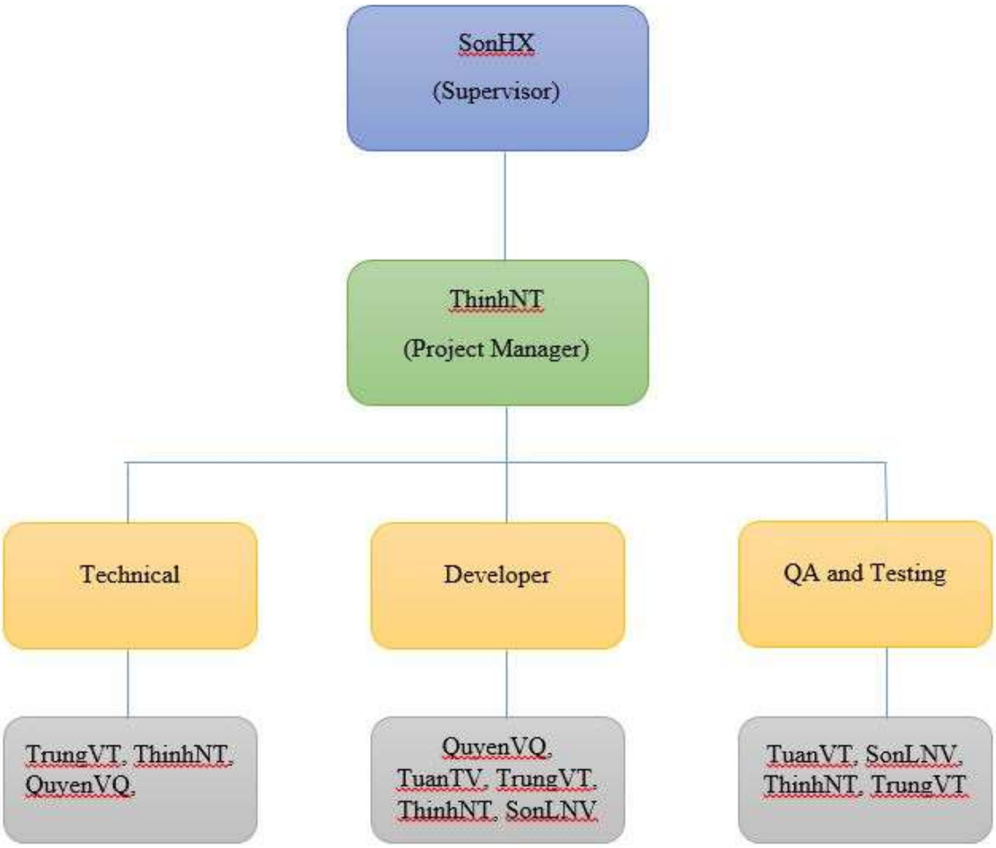


Figure 9 - Project Team Member

2.3 Tool and Techniques:

Hardware

The robot hardware can be classified into the following parts:

- Mechanical Model
- Custom Mainboard
- Inertial sensors : combine gyroscopes and accelerometers
- Actuators: (DC Motor with QEI Encoder)
- Wireless Communication : Bluetooth module
- Addition hardware: - Program/Debug Device

Software

Arduino

- Operating System: Windows 10.
- Hardware Simulation Software: ISIS Proteus 8 Professional.
- Technologies: Arduino, C, C++.
- IDE: Arduino, Sublime Text 3.

Android

- Wireless Communication: Bluetooth module
- Technologies: Java, Android.
- IDE: Android Studio.

3. Project Management Plan:

3.1 Task:

	Indicators	Task Mode	Task Name	Duration	Start	Finish	Pred	Resource Names	Early Start
1	✓		**Initiating**	**5 days**	**Tue 3/22/16**	**Mon 3/28/16**			**Tue 3/22/16**
2	✓		**Develop Project Charter**	**5 days**	**Tue 3/22/16**	**Mon 3/28/16**			**Tue 3/22/16**
3	✓		Kich off meeting	1 day	Tue 3/22/16	Tue 3/22/16		ThinhNT	Tue 3/22/16
4	✓		Identify Goals and Objectives	1 day	Tue 3/22/16	Tue 3/22/16		QuyenVQ,Son	Tue 3/22/16
5	✓		Specify roles and responsibilities	2 days	Wed 3/23/16	Thu 3/24/16	4	QuyenVQ	Wed 3/23/16
6	✓		Estimate project budget	2 days	Wed 3/23/16	Thu 3/24/16	4	SonLNV	Wed 3/23/16
7	✓		Identify main project success criteria	2 days	Wed 3/23/16	Thu 3/24/16	4	TrungVT,TuanT	Wed 3/23/16
8	✓		Develop Project Charter	2 days	Fri 3/25/16	Mon 3/28/16	7	ThinhNT	Fri 3/25/16
9			**Planning**	**16 days**	**Mon 4/4/16**	**Sun 4/24/16**	**1**		**Mon 4/4/16**
10			**Project Integration Management**	**1 day**	**Mon 4/4/16**	**Mon 4/4/16**	**8**		**Mon 4/4/16**
11			Develop project management plan	1 day	Mon 4/4/16	Mon 4/4/16	8	ThinhNT	Mon 4/4/16
12			**Set up Project Enviroment**	**1 day**	**Mon 4/4/16**	**Mon 4/4/16**			**Mon 4/4/16**
13			Set up necessary tools	1 day	Mon 4/4/16	Mon 4/4/16		ThinhNT	Mon 4/4/16
14			**Project Scope Management**	**3 days**	**Tue 4/5/16**	**Thu 4/7/16**	**12**		**Tue 4/5/16**
15			Define Project Scope	3 days	Tue 4/5/16	Thu 4/7/16	12	ThinhNT	Tue 4/5/16
16			Collect requirements	3 days	Tue 4/5/16	Thu 4/7/16	12	ThinhNT	Tue 4/5/16
17			**Project Time Management**	**6 days**	**Thu 4/7/16**	**Thu 4/14/16**	**14**		**Thu 4/7/16**
18			Build Work Breakdown	5 days	Thu 4/7/16	Thu 4/14/16	14	ThinhNT	Thu 4/7/16
19			Specify Deliverables	5 days	Fri 4/8/16	Thu 4/14/16		ThinhNT	Fri 4/8/16
20			Develop Resource Plans	5 days	Fri 4/8/16	Thu 4/14/16	14	ThinhNT	Fri 4/8/16
21			**Project Cost Management**	**7 days**	**Mon 4/11/16**	**Tue 4/19/16**	**14**		**Mon 4/11/16**
22			Estimate cost	3 days	Mon 4/11/16	Wed 4/13/16	14	ThinhNT	Mon 4/11/16
23			Determine budget	3 days	Mon 4/11/16	Wed 4/13/16	14	ThinhNT	Mon 4/11/16
24			**Project Quality Management**	**5 days**	**Mon 4/11/16**	**Fri 4/15/16**	**14**		**Mon 4/11/16**
25			Create quality criteria	2 days	Mon 4/11/16	Tue 4/12/16	14	ThinhNT	Mon 4/11/16
26			Document Quality Management Plan	3 days	Wed 4/13/16	Fri 4/15/16	25	ThinhNT	Wed 4/13/16
27			**Project Risk Management**	**11 days**	**Sun 4/10/16**	**Fri 4/22/16**	**14**		**Sun 4/10/16**
28			Identify risks	8 days	Sun 4/10/16	Wed 4/20/16	14	SonLNV	Sun 4/10/16
29			Plan risk management	2 days	Thu 4/21/16	Fri 4/22/16	28	SonLNV	Thu 4/21/16
30			**Executing**	**76 days**	**Mon 4/25/16**	**Sat 8/6/16**	**9**		**Mon 4/25/16**
31			**Robot**	**30 days**	**Mon 4/25/16**	**Fri 6/3/16**	**9**		**Mon 4/25/16**
32			**Acquire project team**	**1 day**	**Mon 4/25/16**	**Mon 4/25/16**	**9**	**ThinhNT**	**Mon 4/25/16**
33			Project staff assignments	1 day	Mon 4/25/16	Mon 4/25/16	9	ThinhNT	Mon 4/25/16
34			Resource calendars	1 day	Mon 4/25/16	Mon 4/25/16	9	ThinhNT	Mon 4/25/16
35			**Direct and manage project execution**	**23 days**	**Tue 4/26/16**	**Thu 5/26/16**	**32**		**Tue 4/26/16**
36			**Make circuit design**	**1 day**	**Tue 4/26/16**	**Tue 4/26/16**	**34**	**ThinhNT**	**Tue 4/26/16**
37			**Coding and run simulation**	**15 days**	**Wed 4/27/16**	**Tue 5/17/16**	**36**		**Wed 4/27/16**
38			Main program	3 days	Wed 4/27/16	Fri 4/29/16	36	ThinhNT	Wed 4/27/16
39			Control two wheel	3 days	Wed 4/27/16	Fri 4/29/16	36	QuyenVQ	Wed 4/27/16
40			Read sensor	3 days	Wed 4/27/16	Fri 4/29/16	36	TuanTV	Wed 4/27/16
41			Bluetooth connection Mobile	5 days	Wed 4/27/16	Tue 5/3/16	36	SonLNV,Trung	Wed 4/27/16
42			Match to balance simulation	5 days	Mon 5/2/16	Fri 5/6/16	38	QuyenVQ,Thir	Mon 5/2/16
43			Make move (forward, backward, left, right...)	7 days	Mon 5/9/16	Tue 5/17/16	42	QuyenVQ,Son	Mon 5/9/16
44			**Testing, Debug and fix bugs**	**5 days**	**Wed 5/18/16**	**Tue 5/24/16**	**37**	**QuyenVQ,Son**	**Wed 5/18/16**
45			**Run in fact**	**2 days**	**Wed 5/25/16**	**Thu 5/26/16**	**44**	**QuyenVQ,Son**	**Wed 5/25/16**

46	Perform quality assurance	3 days	Fri 5/27/16	Tue 5/31/16	35		Fri 5/27/16
47	Review coding	2 days	Fri 5/27/16	Mon 5/30/16	44	ThinhNT,Trung	Fri 5/27/16
48	Integration Test	2 days	Fri 5/27/16	Mon 5/30/16	44	QuyenVQ,Sonl	Fri 5/27/16
49	Acquire project team	1 day	Tue 5/31/16	Tue 5/31/16	48	ThinhNT	Tue 5/31/16
50	Bicycle	46 days	Mon 6/6/16	Sat 8/6/16	31		Mon 6/6/16
51	Acquire project team	1 day	Mon 6/6/16	Mon 6/6/16	9		Mon 6/6/16
52	Project staff assignments	1 day	Mon 6/6/16	Mon 6/6/16	9	ThinhNT	Mon 6/6/16
53	Resource calendars	1 day	Mon 6/6/16	Mon 6/6/16	9		Mon 6/6/16
54	Direct and manage project execution	40 days	Tue 6/7/16	Mon 8/1/16	51		Tue 6/7/16
55	Make circuit design	1 day	Tue 6/7/16	Tue 6/7/16	53	ThinhNT	Tue 6/7/16
56	Coding and run simulation	32 days	Wed 6/8/16	Thu 7/21/16	55		Wed 6/8/16
57	Main program	10 days	Wed 6/8/16	Tue 6/21/16	55	ThinhNT	Wed 6/8/16
58	Control two wheel	10 days	Wed 6/8/16	Tue 6/21/16	55	QuyenVQ	Wed 6/8/16
59	Read sensor	10 days	Wed 6/8/16	Tue 6/21/16	55	TuanTV	Wed 6/8/16
60	Bluetooth connection Mobile	10 days	Wed 6/8/16	Tue 6/21/16	55	SonLNV,Trung'	Wed 6/8/16
61	Match to balance simulation	12 days	Wed 6/22/16	Thu 7/7/16	57	QuyenVQ,Sonl	Wed 6/22/16
62	Make move (forward, backward, left, right...)	10 days	Fri 7/8/16	Thu 7/21/16	61	QuyenVQ,Sonl	Fri 7/8/16
63	Testing, Debug and fix bugs	5 days	Fri 7/22/16	Thu 7/28/16	56	QuyenVQ,Sonl	Fri 7/22/16
64	Run in fact	2 days	Fri 7/29/16	Mon 8/1/16	63	QuyenVQ,Sonl	Fri 7/29/16
65	Perform quality assurance	3 days	Tue 8/2/16	Thu 8/4/16	54		Tue 8/2/16
66	Review coding	2 days	Tue 8/2/16	Wed 8/3/16	63	ThinhNT,Trung	Tue 8/2/16
67	Integration Test	2 days	Tue 8/2/16	Wed 8/3/16	63	QuyenVQ,Sonl	Tue 8/2/16
68	Acquire project team	1 day	Thu 8/4/16	Thu 8/4/16	67	ThinhNT	Thu 8/4/16
69	Monitoring and Controlling	10 days	Mon 8/8/16	Fri 8/19/16	30		Mon 8/8/16
70	Monitor and control risks	10 days	Mon 8/8/16	Fri 8/19/16	30		Mon 8/8/16
71	Risk register updates	2 days	Mon 8/8/16	Tue 8/9/16		ThinhNT	Mon 8/8/16
72	Resolve risks	8 days	Wed 8/10/16	Fri 8/19/16	71	QuyenVQ,Sonl	Wed 8/10/16
73	Control Schedule						Mon 8/8/16
74	Manage developers					ThinhNT	Mon 8/8/16
75	Control delivery time					ThinhNT	Mon 8/8/16
76	Control time of coding					ThinhNT	Mon 8/8/16
77	Perform quality control						Mon 8/8/16
78	Create test case					SonLNV,TuanT	Mon 8/8/16
79	Manage tester					QuyenVQ,Thir	Mon 8/8/16
80	Execute process of testing					ThinhNT,Trung	Mon 8/8/16
81	Control cost						Mon 8/8/16
82	Project management cost update					ThinhNT	Mon 8/8/16
83	Closing	6 days	Mon 8/22/16	Sat 8/27/16	30		Mon 8/22/16
84	Close project	6 days	Mon 8/22/16	Sat 8/27/16			Mon 8/22/16
85	Final Project Document	5 days	Mon 8/22/16	Fri 8/26/16		SonLNV	Mon 8/22/16
86	Review and Recognize Team Performance	1 day	Sat 8/27/16	Sat 8/27/16		ThinhNT	Sat 8/27/16

Figure 10- Work Breakdown Structure

Refer file Amazing_Bike_Plan.mpp to view details.

3.2 Meeting minute:

An example of group's meeting minute during the time executed project:

Meeting/Project Name:	Amazing Bike		
Date of Meeting:	03/22/2016	***Time: (Type)***	2 hours
Facilitator:	ThinhNT	***Location:***	Library of FPT University
Note Taker:	TrungVT		
1. Meeting Objective:			

<table>
<tr><td>Meeting/Project Name:</td><td colspan="3">Amazing Bike</td></tr>
<tr><td>Date of Meeting:</td><td>03/22/2016</td><td>Time: (Type)</td><td>2 hours</td></tr>
<tr><td>Facilitator:</td><td>ThinhNT</td><td rowspan="2">Location:</td><td rowspan="2">Library of FPT University</td></tr>
<tr><td>Note Taker:</td><td>TrungVT</td></tr>
<tr><td colspan="4">Kick off and create Project Charter.</td></tr>
</table>

<table>
<tr><td colspan="4">2. Attendance</td></tr>
<tr><th>Name</th><th>Roles</th><th>E-mail</th><th>Phone</th></tr>
<tr><td>Nguyễn Trọng Thịnh</td><td>Project Manager</td><td>ThinhNTSE03170@fpt.edu.vn</td><td>0167 2559 886</td></tr>
<tr><td>Vũ Tiến Trung</td><td>Technical Leader</td><td>TrungVTSE03102@fpt.edu.vn</td><td>093 4217 251</td></tr>
<tr><td>Trần Văn Tuấn</td><td>QA Leader</td><td>TuanTVSE03152@fpt.edu.vn</td><td>097 3322 413</td></tr>
<tr><td>Vũ Quang Quyền</td><td>Developer</td><td>QuyenVQSE03237@fpt.edu.vn</td><td>0167 4092 091</td></tr>
<tr><td>Lưu Ngọc Việt Sơn</td><td>Developer</td><td>SonLNVSE03711@fpt.edu.vn</td><td>091 2872 336</td></tr>
<tr><td colspan="4">3. Content:</td></tr>
<tr><td colspan="4">1. Kick off meeting.
2. Identify Goals and Objectives.
3. Specify roles and responsibilities.
4. Estimate project budget.
5. Identify main project success criteria.
6. Develop Project Charter.
7. Assign mission for each member.
8. Set up time for next meeting.</td></tr>
</table>

Table 3 - Meeting Minutes Template

3.3 Coding convention:

The following rules follow:

- The standard rules for developing application using java programmer language to build Android application (https://source.android.com/source/code-style.html).
- API Style Guide for Arduino (https://www.arduino.cc/en/Reference/StyleGuide).

3.4 Risk manage plan:

Risk ID	Avoiding plan	Fallback plan	Contingency plan	Schedule
R1	- PM need contact to stakeholder so that he can aware of changes ASAP.	- Should prepare a list of other suppliers.	- Use chips, modules from other suppliers with similar functions.	- Project manager. - Contacts, regularly updated information from the suppliers.
R2	- Make budget plan detail at the first step of project.	- Modify budget plan to avoid over budget.	- Minimize expense. - Get money from sponsor.	- Project manager. - Control budget frequency.
R3	- PM should estimate workforce needed for the project and able reserve members if needed.	- Assign clearly jobs based on the capacity of the group members.	- A few excellent members may undertake other work more to complete the project.	- Project manager. - Estimate workforce needed for the project.
R4	- Learning technology clearly before starting project.	- Finding supporting from expert.	- Limit related function.	- Technical Leader and Project manager. - Contact to expert as soon as possible after an issues occurs.
R5	- Should create a schedule in which tasks are divided to suitable members.	- Check deadlines regularly and motivate members to meet the deadline.	- Reprimand, reprove, change member if necessary.	- Project manager and QA leader. - Check deadline regularly.
R6	- Training members.	- Support members.	- Changing members.	- Technical Leader and members. - Training members in 1 week after kick off project. - Supporting members: member has technical issues.

R7	- Hiring member appropriately	- Setting common rule	Meeting to identify common goal	- Project manager and QA leader - Organizing daily and weekly meeting
R8	- PM should discuss with developer team to find solution.	- Testing new technologies, modern equipment	- Finding supporting from expert	- Project manager and team member. - Organizing important meeting.

Table 4- Risk Management Plan

3.5 Communication plan:

- ***Weekly meeting schedule***: By using Iterative and Incremental Process Model, we have weekly meeting to review, update and solve all problems that team has to face during working time. It's often occurred on Tuesday or Thursday at the library of FPT University.

  Beside of discussing some current issues, this is team time for all members can fix problems together. On the other hand, technical leader can help others with technical problem and keep track working process.
- ***Unscheduled meeting***: If someone has an important problem want to be solved immediately, he/she can propose to Project Manager to hold urgent meeting.
- ***Communication channel***: Our main communication channels are physical meeting, email. However, we often have phone call or skype calling meeting.

III. Software requirement specification (SRS):

1. Purpose:

This chapter outlines functional and non – functional requirements of our website. It also provides some format constraints in common requirements and project success criteria. All members will work (design, code, test) based on the information provided in this chapter.

2. External Interface Requirements:

2.1 User Interface:

The interface must be designed to be satisfied the following requirement:

- The interface is divided by tabs, which will allow users to easily switch between different parts of the program.
- Be simple and user-friendly.
- Meet all the main functions and easily to identify each of functions.
- Use obvious icons to avoiding misunderstanding.

2.2 Hardware Interface:

The hardware interfaces the robot using must be designed to be satisfied the following requirement:

- Low-cost hardware module.
- Easy to consume.
- Easy to replace for maintenance.

2.3 Communication Protocols:

With the robot and the android application, we must use Serial Port Protocol (SPP) to connect each other, via Bluetooth Module of PC and HC-05 Bluetooth Module on the robot.

With the self-balance bike and the android application, we use radio wave to connect them because the distance from the mobile to the bike is far.

3. Functional Requirements:

3.1 Use Case Diagram:

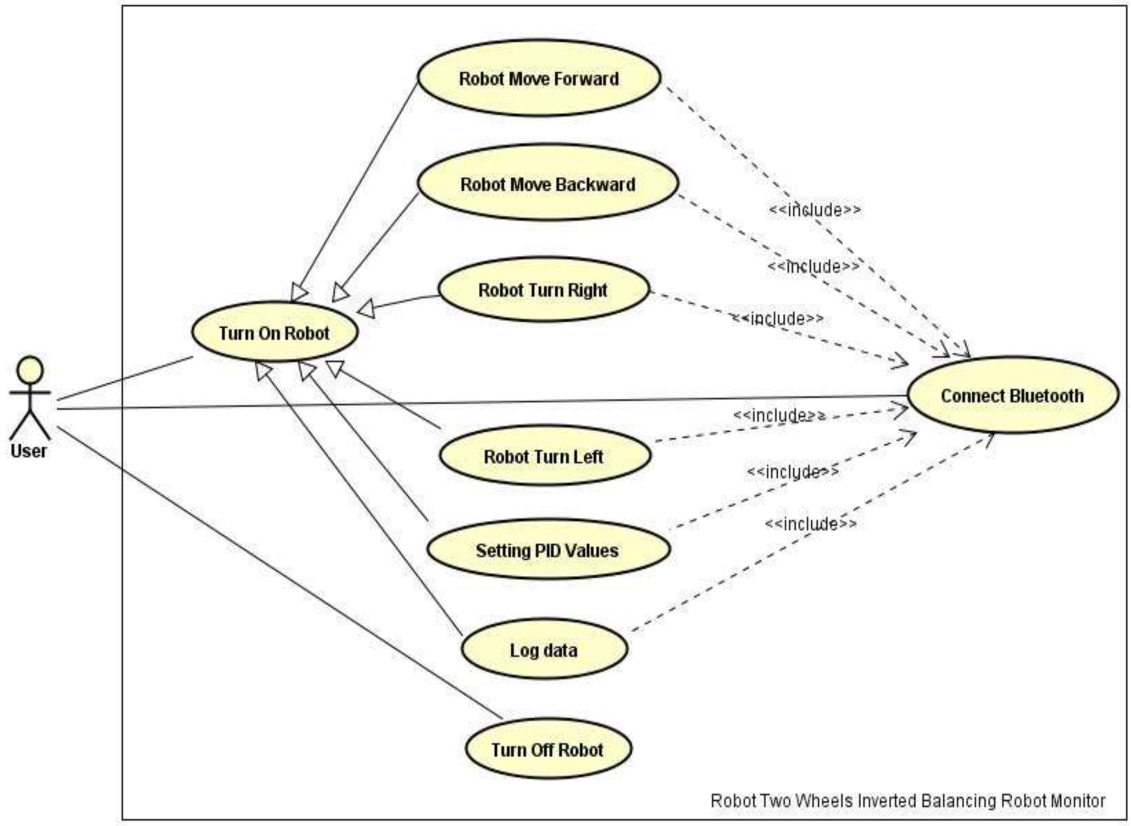


Figure 11 – Use case Diagram Robot Two Wheels

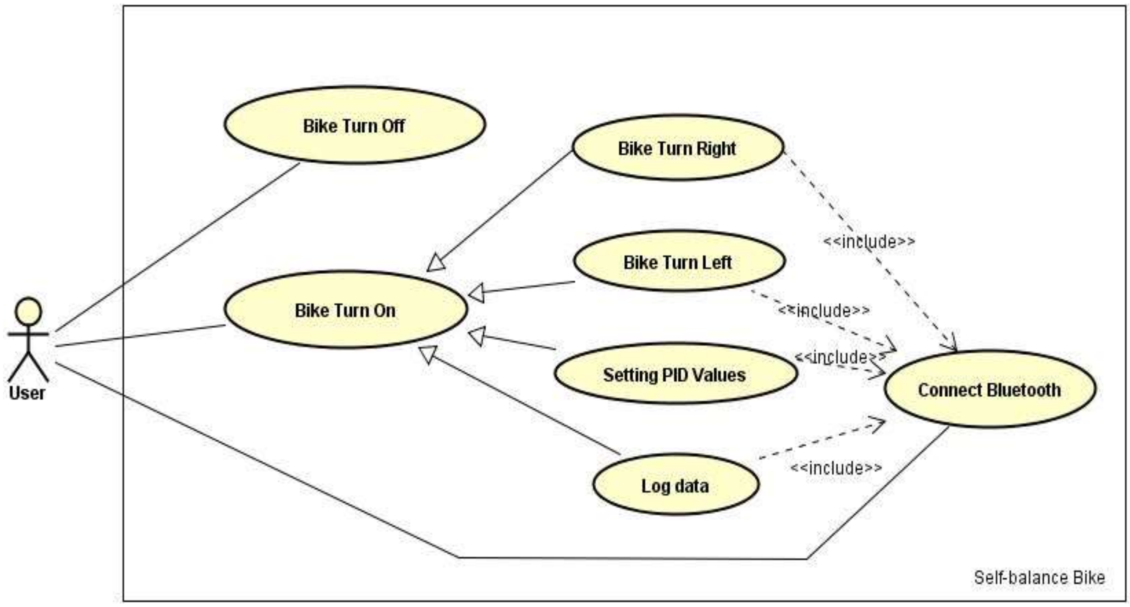


Figure 12 - Use case Self-balance Bike

3.2 Use Case Lists:

3.2.1 Robot two wheels:

No	Function No	Function Name	Description
1	R2W01	Robot turn On.	Start the robot, robot can keep self-balance state.
2	R2W02	Robot turn Off.	Turn off the motor, robot stops run.
3	R2W03	Connect Bluetooth.	Connect the robot with android application via Bluetooth.
4	R2W04	Setting PID values.	Set PID parameters for robot from mobile phone that uses android platform.
5	R2W05	Robot move forward.	Control the robot to move ahead.
6	R2W06	Robot move backward.	Control the robot to move rearwards.
7	R2W07	Robot turn right.	Control the robot to turn right.
8	R2W08	Robot turn left.	Control the robot to turn left.
9	R2W09	Log data	Display the information about angle, PID values on the mobile screen application.

Table 5 - Robot two wheels Use case

3.2.2 Self-Balance Bike:

No	Function No	Function Name	Description
1	SBB01	Connect Bluetooth	Connect the bike with android application via Bluetooth.
2	SBB02	Bike turn On	Start the motor, bicycle auto moving ahead.
3	SBB03	Bike turn Off	Turn off the motor, bicycle stops moving.
4	SBB04	Setting PID values	Set PID parameters automatically that passed from the mobile.
5	SBB05	Bike turn right	Control the bicycle to turn right.
6	SBB06	Bike turn left	Control the bicycle to turn left.
7	SBB07	Log data	Display the information about angle, PID values on the mobile screen application.

Table 6 - Self-Balance Bike Use case

3.3 Use Cases:

3.3.1 Robot turn on:

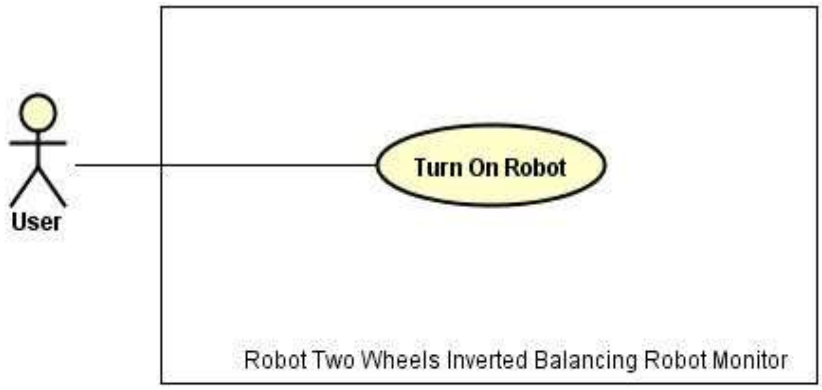


Figure 13- Use case Turn on Robot

Use Case ID	**R2W01**	**Use Case Name**		Robot turn On	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	User start the robot.				
Precondition	N/A				
Trigger	User turn on the starting switch on the robot.				
Post-Condition	The robot is started and can keep self-balance standing state.				

Main flows

Step	***Actor***	***Action***
1	User	User press starting button on the robot.
2	System Response	Robot is ON.

Alternative flows

N/A

Exceptions

No	**Actor Action**	**System Response**
1	• The robot hasn't yet fully fitted mains (power).	The robot doesn't start.

	• User presses the button	
Business Rules		
#	***Rule Description***	

3.3.2 Robot turn off:

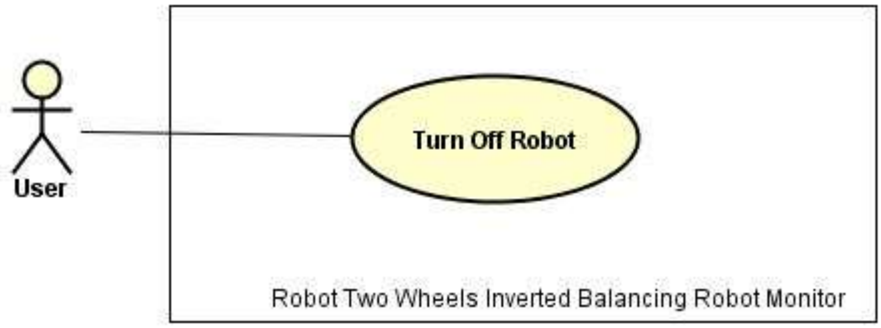


Figure 14 - Use case Turn off Robot

Use Case ID	**R2W02**	**Use Case Name**		Robot turn Off	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	User stops running the robot.				
Precondition	Robot is ON.				
Trigger	User turn off the starting switch on the robot.				
Post-Condition	Robot is OFF.				
Main flows					
Step	***Actor***	***Action***			
1	User	User turn off the starting switch on the robot.			
2	System Response	Robot is OFF and fall immediately.			
Alternative flows					
N/A					
Exceptions:					

No	Actor Action	System Response
N/A		
Business Rules		
#	***Rule Description***	

3.3.3 Connect Robot with Android application via Bluetooth:

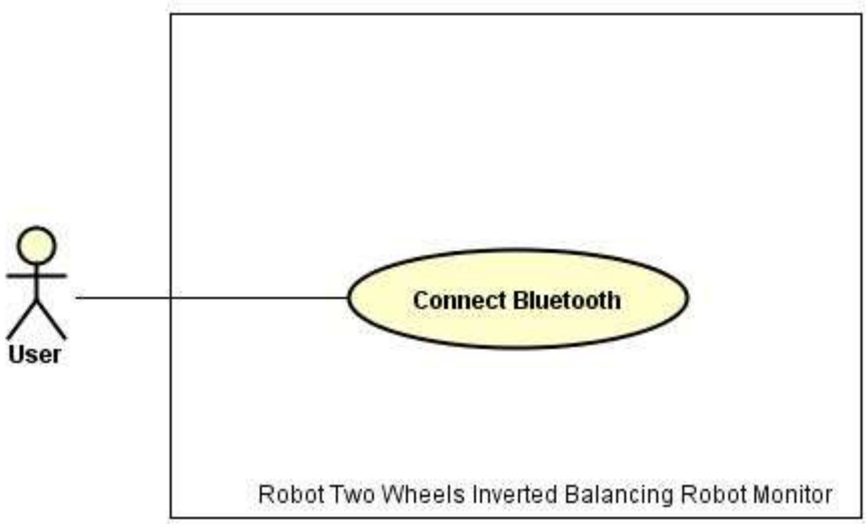


Figure 15 - Use case Connect Bluetooth

Use Case ID	**R2W03**	**Use Case Name**		Connect Bluetooth	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	User connects the robot to the mobile via Bluetooth connection.				
Precondition	• Robot is ON. • Android application is not connected to robot.				
Trigger	User touches “Connect” button on the screen interface of android application.				
Post-Condition	• Android app connected to the robot. • Android app displays “Connected” or “Disconnected” on the screen.				

Main flows		
Step	***Actor***	***Action***
1	User	User open Android application on smart phone.
2	System Response	Application is appeared. The welcome screen is displayed.
3	User	Click on "Connect" button on display screen.
4	System Response	Settings panel is opened, port name list and baud rate list is shown.
5	User	User click on "Connect Now" button
6	System Response	• Android application connect to selected port name and baud rate. • Android application will show status "Connected" on the screen.
Alternative flows		
N/A		
Exceptions:		
No	**Actor Action**	**System Response**
1	• Choose other port name (is not the robot Bluetooth port name). • Click "Connect Now" button.	Appear message: "Cannot establish the connection! Please check then try again!"
2	• Turn off robot. • Click "Connect Now" button.	Appear message: "Cannot establish the connection! Please check then try again!"
3	The PC does not have any available COM port.	Appear message "There are no port in your PC, Please Add a Bluetooth device and try again!"
Business Rules		
#	***Rule Description***	

3.3.4 Setting PID values:

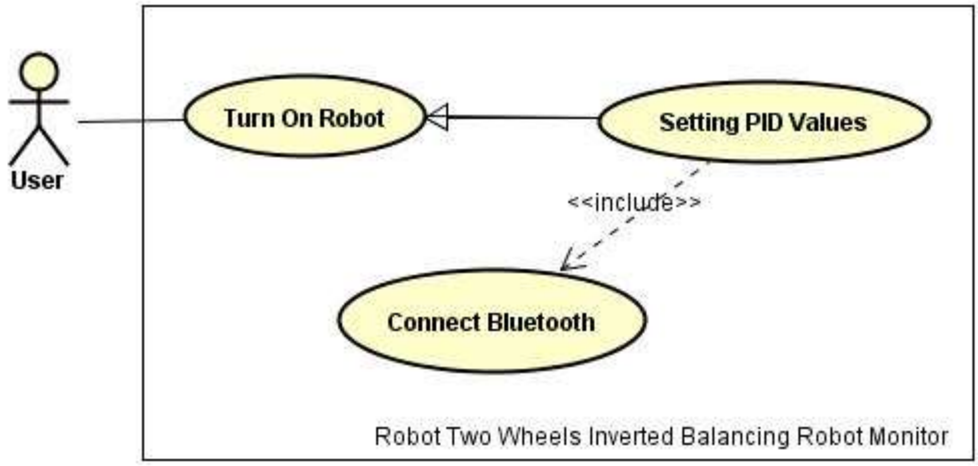


Figure 16 - Use case Setting PID values for robot

Use Case ID	**R2W04**	**Use Case Name**		Setting PID values	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	This use case allow user to set Proportional, Integral, Derivative (PID)				
Precondition	• Robot is ON. • Android application is connected to robot. • No other function is executing.				
Trigger	User enters the values into Kp, Ki, Kd at "Setting PID" on the screen.				
Post-Condition	• **Success**: The confirm notification is shown to notice the success of setting PID values. • **Fail**: Failure message is shown.				

Main flows

Step	***Actor***	***Action***
1	User	User fills values to Proportional, Integral, Derivative textboxes on the display screen.
2	User	Click on "Send" button on display screen.
4	System Response	• Robot change the activity based on the new PID values. • Display new PID values on the screen of the smart phone.

Alternative flows

N/A

Exceptions		
No	**Actor Action**	**System Response**
1	Application is disconnected to the robot.	• Application return to disconnected status • Display error message "Lost connection with the robot. Please check the connection and try again".
2	User leaves Proportional, Integral, Derivative textboxes blank or fills invalid values in it.	Display error message "Please check and fill value of PID in the textboxes".
Business Rules		
#	***Rule Description***	

3.3.5 Robot move forward:

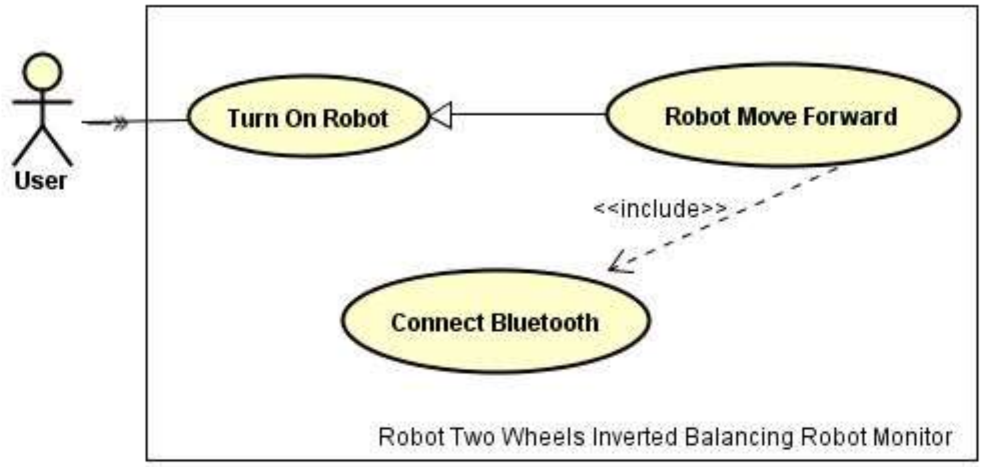


Figure 17 - Use case Robot move forward

Use Case ID	**R2W05**	**Use Case Name**		Robot move forward	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	User controls the robot to move ahead by mobile phone.				

Precondition		• Robot is ON • Android application is connected to robot. • No other function is executing.
Trigger		User click on "Forward" button on the screen interface of Android app.
Post-Condition		Robot stop when user press "Stop" button.
Main flows		
Step	***Actor***	***Action***
1	User	User open Android application on smart phone.
2	System Response	Application is appeared. The welcome screen is displayed.
3	User	User press "Forward" button on main screen.
4	System Response	Robot is moving forward.
5	User	User press "Stop" button on the main screen.
6	System Response	Robot stop moving and soon stands upright.
Alternative flows		
N/A		
Exceptions		
No	**Actor Action**	**System Response**
1	• User turns off the robot. • User press "Forward" button on main screen.	• Application return to **disconnected** status • Display error message "Lost connection with the robot. Please check the connection and try again."
Business Rules		
#	***Rule Description***	

3.3.6 Robot move backward:

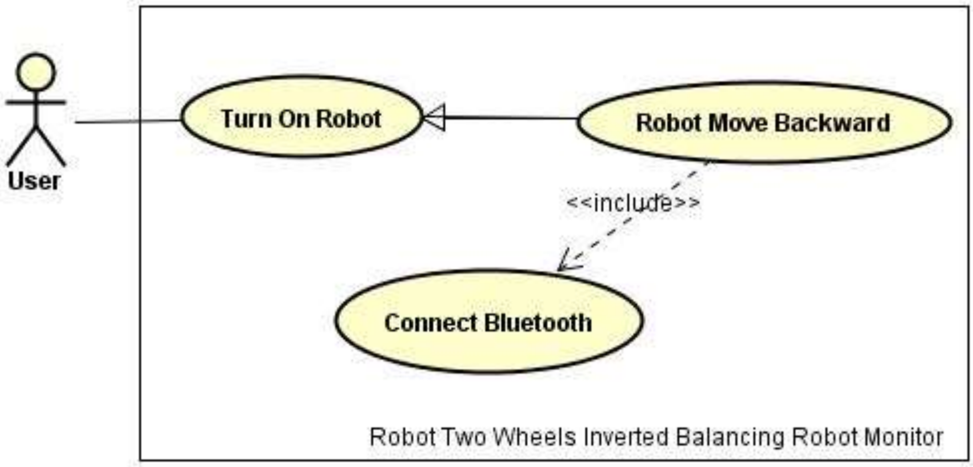


Figure 18 - Use case Robot move backward

Use Case ID	**R2W06**		**Use Case Name**		Robot move backward	
Author	**SonLNV**		**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User					
Description	User controls the robot moving rearward by mobile phone.					
Precondition	• Robot is ON • Android application is connected to robot. • No other function is executing.					
Trigger	User click on "Backward" button on the screen interface of Android app.					
Post-Condition	Robot stop when user press "Stop" button.					
Main flows						
Step	***Actor***	***Action***				
1	User	User open Android application on smart phone.				
2	System Response	Application is appeared. The welcome screen is displayed.				
3	User	User press "Backward" button on the main screen.				
4	System Response	Robot is moving backward.				
5	User	User press "Stop" button on the main screen.				
6	System Response	Robot stop moving and soon stands upright.				
Alternative flows						

N/A		
Exceptions		
No	**Actor Action**	**System Response**
1	• User turns off the robot. • User press "Backward" button on main screen.	• Application return to **disconnected** status • Display error message "Lost connection with the robot. Please check the connection and try again."
Business Rules		
#	***Rule Description***	

3.3.7 Robot turn right:

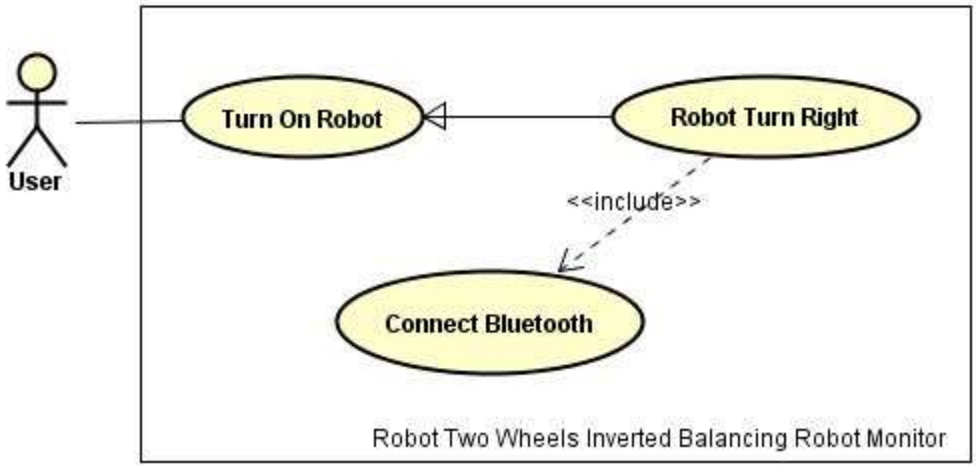


Figure 19 - Use case Robot turn right

Use Case ID	**R2W07**	**Use Case Name**		Robot turn right	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	User controls the robot to turn right by mobile phone.				
Precondition	• Robot is ON • Android application is connected to robot. • No other function is executing.				

Trigger		User press "TurnRight" button on the screen interface of android app.
Post-Condition		Robot stop when user press "Stop" button.
Main flows		
Step	***Actor***	***Action***
1	User	User open Android application on smart phone.
2	System Response	Application is appeared. The welcome screen is displayed.
3	User	User press "TurnRight" button on the main screen.
4	System Response	Robot is turning right.
5	User	User press "Stop" button on the main screen.
6	System Response	Robot stop turning and soon stands upright.
Alternative flows		
N/A		
Exceptions		
No	**Actor Action**	**System Response**
1	• User turns off the robot. • User press "TurnRight" button.	• Application return to **disconnected** status • Display error message "Lost connection with the robot. Please check the connection and try again."
Business Rules		
#	***Rule Description***	

3.3.8 Robot turn left:

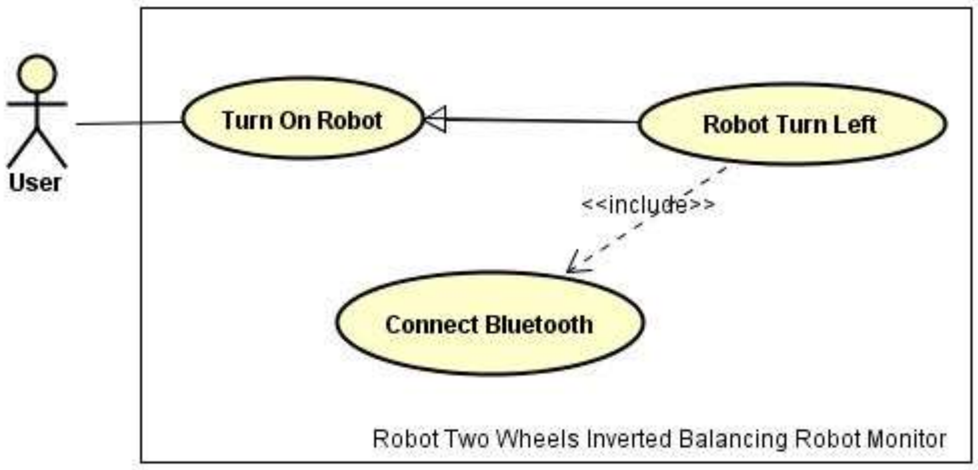


Figure 20 - Use case Robot turn left

Use Case ID	**R2W08**	**Use Case Name**		Robot turn left	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	User controls the robot to turn left by mobile phone.				
Precondition	• Robot is ON • Android application is connected to robot. • No other function is executing.				
Trigger	User press “TurnLeft” button on the screen interface of android app.				
Post-Condition	Robot stop when user press “Stop” button.				

Main flows

Step	***Actor***	***Action***
1	User	User open Android application on smart phone.
2	System Response	Application is appeared. The welcome screen is displayed.
3	User	User press “TurnLeft” button on the main screen.
4	System Response	Robot is turning left.
5	User	User press “Stop” button on the main screen.
6	System Response	Robot stop turning and soon stands upright.

Alternative flows		
N/A		
Exceptions		
No	**Actor Action**	**System Response**
1	• User turns off the robot. • User press "TurnLeft" button.	• Application return to **disconnected** status • Display error message "Lost connection with the robot. Please check the connection and try again."
Business Rules		
#	***Rule Description***	

3.3.9 Log data:

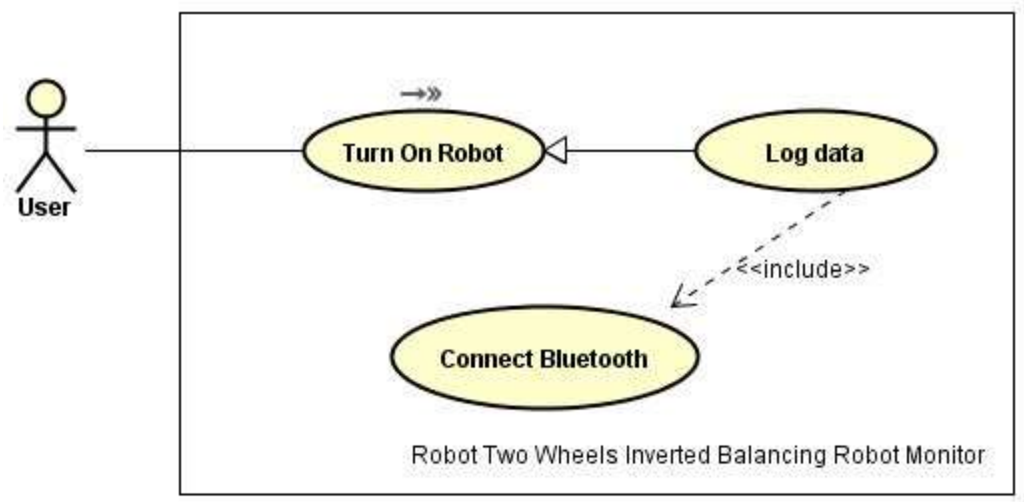


Figure 21 - Use case Robot log data

Use Case ID	**R2W09**	**Use Case Name**		Log data	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	System				
Description	System display information about the operating status, the PID values to the screen.				
Precondition	• Robot is ON • Android application is connected to robot. • No other function is executing.				

Trigger		N/A
Post-Condition		The information is displayed on the screen of the mobile phone.
Main flows		
Step	***Actor***	***Action***
1	User	User open Android application on smart phone.
2	System Response	Application is appeared. The welcome screen is displayed.
3	User	User control the robot is operational.
4	System Response	System display information about PID values, the operating status and the graph of the balance angle and the graph of the PWM (Pulse Width Modulation) values on screen of mobile phone.
Alternative flows		
N/A		
Exceptions		
No	**Actor Action**	**System Response**
Business Rules		
#	***Rule Description***	

3.3.10 Connect Bluetooth to bike:

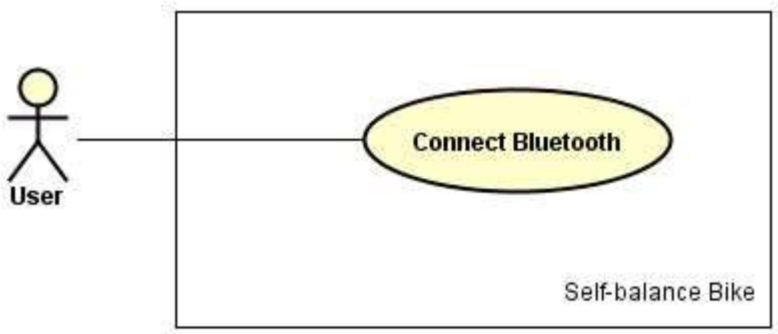


Figure 22 - Use case Connect Bluetooth of Bike

<table>
<tr><td colspan="2">Use Case ID</td><td colspan="2">SBB01</td><td colspan="2">Use Case Name</td><td colspan="2">Connect Bluetooth</td></tr>
<tr><td colspan="2">Author</td><td colspan="2">SonLNV</td><td>Version</td><td>0.1</td><td>Date</td><td>07/11/2016</td></tr>
<tr><td colspan="2">Actor</td><td colspan="6">User</td></tr>
<tr><td colspan="2">Description</td><td colspan="6">User connects the bike to the mobile via Bluetooth connection.</td></tr>
<tr><td colspan="2">Precondition</td><td colspan="6">• Robot is ON.
• Android application is not connected to bike.</td></tr>
<tr><td colspan="2">Trigger</td><td colspan="6">User touches “Connect” button on the screen interface of android application.</td></tr>
<tr><td colspan="2">Post-Condition</td><td colspan="6">• Android app connected to the bike.
• Android app displays “Connected” or “Disconnected” on the screen.</td></tr>
<tr><td colspan="8">Main flows</td></tr>
<tr><td>Step</td><td>Actor</td><td colspan="6">Action</td></tr>
<tr><td>1</td><td>User</td><td colspan="6">User open Android application on smart phone.</td></tr>
<tr><td>2</td><td>System Response</td><td colspan="6">Application is appeared. The welcome screen is displayed.</td></tr>
<tr><td>3</td><td>User</td><td colspan="6">Click on “Connect” button on display screen.</td></tr>
<tr><td>4</td><td>System Response</td><td colspan="6">Settings panel is opened, port name list and baud rate list is shown.</td></tr>
<tr><td>5</td><td>User</td><td colspan="6">User click on “Connect Now” button</td></tr>
<tr><td>6</td><td>System Response</td><td colspan="6">• Android application connect to selected port name and baud rate.
• Android application will show status “Connected” on the screen.</td></tr>
<tr><td colspan="8">Alternative flows</td></tr>
<tr><td colspan="8">N/A</td></tr>
<tr><td colspan="8">Exceptions:</td></tr>
<tr><td>No</td><td colspan="2">Actor Action</td><td colspan="5">System Response</td></tr>
<tr><td>1</td><td colspan="2">• Choose other port name (is not the robot Bluetooth port name).
• Click “Connect Now” button.</td><td colspan="5">Appear message: “Cannot establish the connection! Please check then try again!”</td></tr>
<tr><td>2</td><td colspan="2">• Turn off robot.
• Click “Connect Now” button.</td><td colspan="5">Appear message: “Cannot establish the connection! Please check then try again!”</td></tr>
</table>

3	The PC does not have any available COM port.	Appear message "There are no port in your PC, Please Add a Bluetooth device and try again!"
Business Rules		
#	***Rule Description***	

3.3.11 Bike turn on:

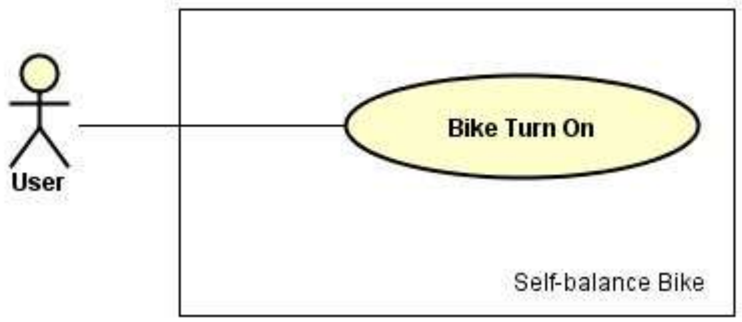


Figure 23 - Use case Bike turn on

Use Case ID	**SBB02**	**Use Case Name**		Bike turn on	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	User starts the bicycle.				
Precondition	N/A				
Trigger	User presses starting button or opens lock on the bicycle.				
Post-Condition	Bicycle starts automatically and runs self-balance.				
Main flows					
Step	***Actor***	***Action***			
1	User	User presses starting button or opens lock on the bicycle.			
2	System Response	• Bike is ON. • Bike go straight and self-balance.			
Alternative flows					
N/A					

Exceptions		
No	**Actor Action**	**System Response**
1	• The bike hasn't yet fully fitted mains (power). • User presses the button.	Bike don't start.
Business Rules		
#	***Rule Description***	

3.3.12 Bike turn off:

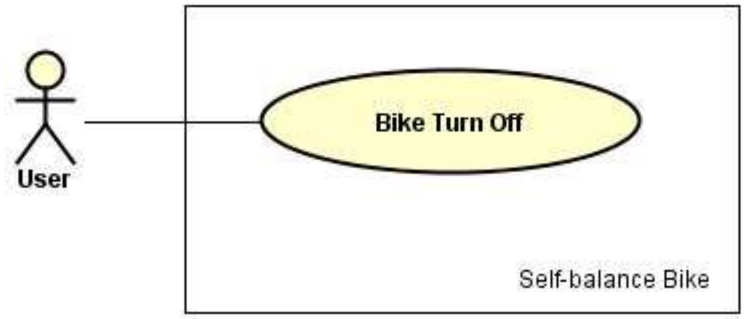


Figure 24 - Use case Bike turn off

Use Case ID	**SBB03**	**Use Case Name**		Bike turn off	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	User stops running Bicycle.				
Precondition	Bike is going straight				
Trigger	User turn off the switch or close the clock on the bike.				
Post-Condition	Bicycle stops moving.				
Main flows					
Step	***Actor***	***Action***			
1	User	User turn off the switch or close the clock on the bike.			
2	System Response	Bike is OFF.			

Alternative flows		
N/A		
Exceptions:		
No	**Actor Action**	**System Response**
N/A		

Business Rules	
#	***Rule Description***

3.3.13 Setting PID values:

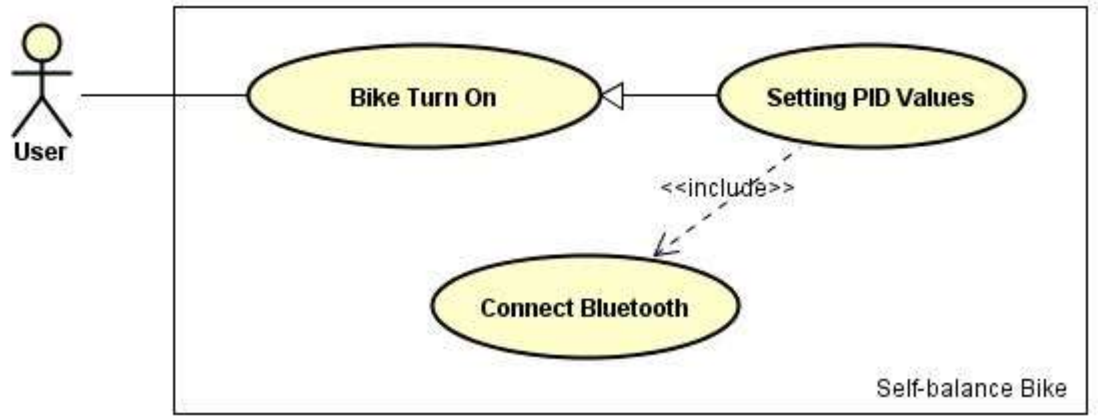


Figure 25 - Use case Setting PID values for bike

Use Case ID	**SSB04**	**Use Case Name**		Setting PID values	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	This use case allow user to set Proportional, Integral, Derivative (PID)				
Precondition	• Bike is ON. • Android application is connected to bike. • No other function is executing.				
Trigger	User enters the values into Kp, Ki,Kd at "Setting PID" on the main screen.				

<table>
<tr><td colspan="2">Post-Condition</td><td>Success: The confirm notification is shown to notice the success of setting PID values.
Fail: Failure message is shown.</td></tr>
<tr><td colspan="3">Main flows</td></tr>
<tr><td>Step</td><td>Actor</td><td>Action</td></tr>
<tr><td>1</td><td>User</td><td>User fills values to Proportional, Integral, Derivative textboxes on the display screen.</td></tr>
<tr><td>2</td><td>User</td><td>Click on “Send” button on display screen.</td></tr>
<tr><td>4</td><td>System Response</td><td>Bike change the activity based on the new PID values.
Display new PID values on the screen of the smart phone.</td></tr>
<tr><td colspan="3">Alternative flows</td></tr>
<tr><td colspan="3">N/A</td></tr>
<tr><td colspan="3">Exceptions</td></tr>
<tr><td>No</td><td>Actor Action</td><td>System Response</td></tr>
<tr><td>1</td><td>Application is disconnected to the bike.</td><td>Application return to disconnected status
Display error message “Lost connection with the bike. Please check the connection and try again”.</td></tr>
<tr><td>2</td><td>User leaves Proportional, Integral, Derivative textboxes blank or fills invalid values in it.</td><td>Display error message “Please check and fill value of PID in the textboxes”.</td></tr>
<tr><td colspan="3">Business Rules</td></tr>
<tr><td>#</td><td colspan="2">Rule Description</td></tr>
<tr><td></td><td colspan="2"></td></tr>
</table>

3.3.14 Bike turn right:

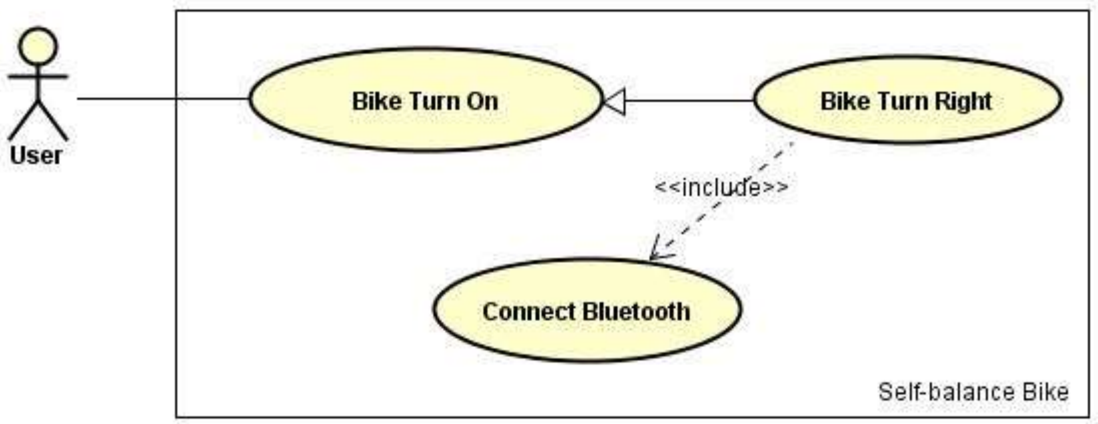


Figure 26 - Use case Bike turn right

<table>
<tr><td colspan="2">Use Case ID</td><td>SBB05</td><td colspan="2">Use Case Name</td><td colspan="2">Bike turn right</td></tr>
<tr><td colspan="2">Author</td><td>SonLNV</td><td>Version</td><td>0.1</td><td>Date</td><td>07/11/2016</td></tr>
<tr><td colspan="2">Actor</td><td colspan="5">User</td></tr>
<tr><td colspan="2">Description</td><td colspan="5">User controls the bicycle to turn right by mobile phone.</td></tr>
<tr><td colspan="2">Precondition</td><td colspan="5">• Bike is ON
• Android application is connected to bike.
• No other function is executing.</td></tr>
<tr><td colspan="2">Trigger</td><td colspan="5">User press “TurnRight” button on the screen interface of android app.</td></tr>
<tr><td colspan="2">Post-Condition</td><td colspan="5">Bike stop turning and go straight when user release center button.</td></tr>
<tr><td colspan="7">Main flows</td></tr>
<tr><td>Step</td><td>Actor</td><td colspan="5">Action</td></tr>
<tr><td>1</td><td>User</td><td colspan="5">User open Android application on smart phone.</td></tr>
<tr><td>2</td><td>System Response</td><td colspan="5">Application is appeared. The welcome screen is displayed.</td></tr>
<tr><td>3</td><td>User</td><td colspan="5">User press “TurnRight” button on the main screen.</td></tr>
<tr><td>4</td><td>System Response</td><td colspan="5">Bike is turning right.</td></tr>
<tr><td>5</td><td>User</td><td colspan="5">User press “Stop” button on the main screen.</td></tr>
<tr><td>6</td><td>System Response</td><td colspan="5">Bike stop turning and soon go straight and self-balance.</td></tr>
</table>

Alternative flows		
N/A		
Exceptions		
No	**Actor Action**	**System Response**
1	• User turns off the bike. • User press "TurnRight" button.	• Application return to **disconnected** status • Display error message "Lost connection with the bike. Please check the connection and try again."
Business Rules		
#	***Rule Description***	

3.3.15 Bike turn left:

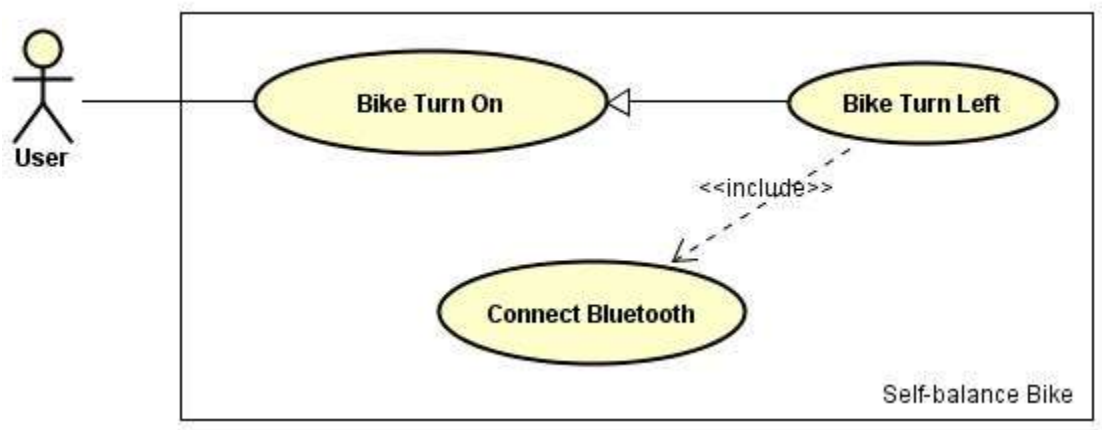


Figure 27 - Use case Bike turn left

Use Case ID	**SSB06**	**Use Case Name**		Bike turn left	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	User				
Description	User controls the bicycle to turn left by mobile phone.				
Precondition	• Bike is ON • Android application is connected to bike. • No other function is executing.				

Trigger		User press “TurnLeft” button on the screen interface of android app.
Post-Condition		Bike stop turning and go straight when user release center button.
Main flows		
Step	***Actor***	***Action***
1	User	User open Android application on smart phone.
2	System Response	Application is appeared. The welcome screen is displayed.
3	User	User press “TurnLeft” button on the main screen.
4	System Response	Bike is turning left.
5	User	User press “Stop” button on the main screen.
6	System Response	Bike stop turning and soon go straight and self-balance.
Alternative flows		
N/A		
Exceptions		
No	**Actor Action**	**System Response**
1	• User turns off the bike. • User press “TurnLeft” button.	• Application return to **disconnected** status • Display error message “Lost connection with the bike. Please check the connection and try again.”
Business Rules		
#	***Rule Description***	

3.3.16 Data log:

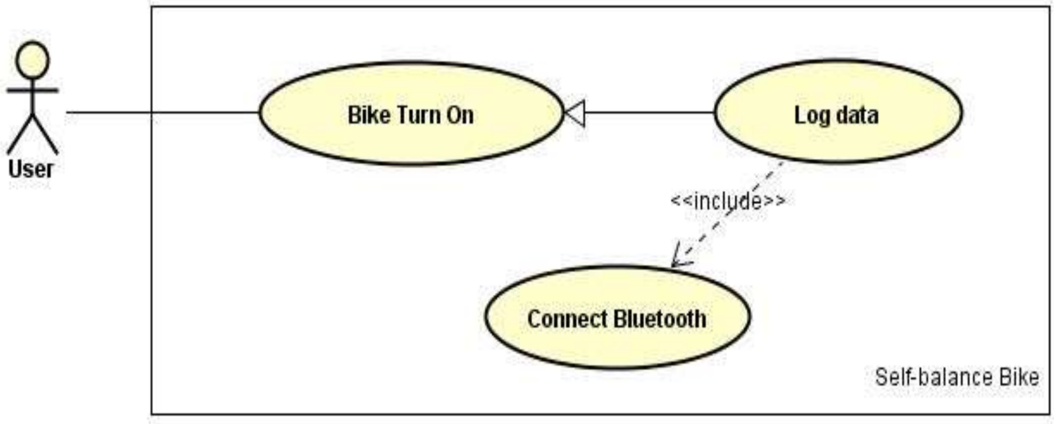


Figure 28 - Use case Log data of Bike

Use Case ID	**SSB07**	**Use Case Name**		Log data	
Author	**SonLNV**	**Version**	**0.1**	**Date**	**07/11/2016**
Actor	System				
Description	System display information about the operating status, the PID values to the screen.				
Precondition	• Robot is ON • Android application is connected to bike. • No other function is executing.				
Trigger	N/A				
Post-Condition	The information is displayed on the screen of the mobile phone.				
Main flows					
Step	***Actor***	***Action***			
1	User	User open Android application on smart phone.			
2	System Response	Application is appeared. The welcome screen is displayed.			
3	User	User control the bike is operational.			
4	System Response	System display information about PID values, the operating status and the graph of the balance angle and the graph of the PWM (Pulse Width Modulation) values on screen of mobile phone.			
Alternative flows					
N/A					
Exceptions					

No	Actor Action	System Response
Business Rules		
#	***Rule Description***	

4. Non-functional Requirements:

4.1 Reliability:

- There is at least 99.9% of user command that will be successfully sent to the robot.
- The robot has to stay balance when it does not receive any disturbance.
- Balancing accuracy of PID algorithms is at least 99 % (The balancing angle is degree) when it does not receive any external disturbance.
- The robot must be able to return to balance state when it receives any external forces.

4.2 Availability:

- User can operate the robot successfully using two wheeled inverted balancing robot monitor whenever the robot is on and available to connect through Bluetooth.

4.3 Security:

- N/A

4.4 Maintainability:

- Flexible mechanical designing to make it easy to replace any robot components.
- The hardware components are all available, and easy to purchase in Viet Nam.
- The system is divided into separate modules (TWIBR, TWIBR Monitor) - The firmware code and software code is easy to maintain and upgrade.

4.5 Portability:

- N/A.

4.6 Performance:

- Robot has to response one user command in less than 0.5 second.
- Robot has to reach balance state in less than 3 seconds after it received an external force.

IV. Research and study:

1. PID:

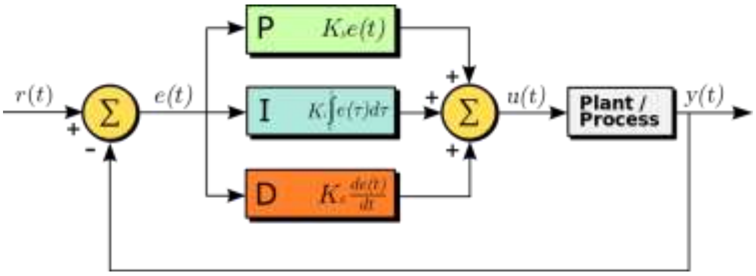


Figure 29 - A block diagram of a PID controller in a feedback loop

Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry and has been universally accepted in industrial control.

The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating conditions and partly to their functional simplicity, which allows engineers to operate them in a simple, straightforward manner.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

Figure 30 - PID algorithm

With Kp, Ki, and Kd, all non-negative, denote the coefficients for the proportional, integral, and derivative terms, respectively (sometimes denoted *P*, *I*, and *D*).

In this model,

- *P* accounts for present values of the error. For example, if the error is large and positive, the control output will also be large and positive.
- *I* accounts for past values of the error. For example, if the current output is not sufficiently strong, error will accumulate over time, and the controller will respond by applying a stronger action.
- *D* accounts for possible future values of the error, based on its current rate of change.

As a PID controller relies only on the measured process variable, not on knowledge of the underlying process, it is broadly applicable. By tuning the three parameters of the model, a PID controller can deal with specific process requirements. The response of the controller can be described in terms of its responsiveness to an error, the degree to which the system overshoots a set point, and the degree of any system oscillation. The use of the PID algorithm does not guarantee optimal control of the system or even its stability.

2. Ziegler-Nichols method:

The Ziegler–Nichols tuning method is a heuristic method of tuning a PID controller. It was developed by John G. Ziegler and Nathaniel B. Nichols. It is performed by setting the

I (integral) and **D** (derivative) gains to zero. The "**P**" (proportional) gain, $\mathbf{K_p}$ is then increased (from zero) until it reaches the ultimate gain $\mathbf{K_u}$, at which the output of the control loop has stable and consistent oscillations. $\mathbf{K_u}$ and the oscillation period $\mathbf{T_u}$ are used to set the P, I, and D gains depending on the type of controller used:

Ziegler–Nichols method			
Control Type	$\mathbf{K_p}$	$\mathbf{T_i}$	**Td**
P	0.5 $\mathbf{K_u}$	-	-
PI	0.45 $\mathbf{K_u}$	$\mathbf{T_u}$ /1.2	-
PD	0.8 $\mathbf{K_u}$	-	$\mathbf{T_u}$ /8
Classic PID	0.6 $\mathbf{K_u}$	$\mathbf{T_u}$ /2	$\mathbf{T_u}$ /8
Pessen Integral Rule	0.7 $\mathbf{K_u}$	$\mathbf{T_u}$ /2.5	3 $\mathbf{T_u}$ /20
Some overshoot	0.33 $\mathbf{K_u}$	$\mathbf{T_u}$ /2	$\mathbf{T_u}$ /3
No overshoot	0.2 $\mathbf{K_u}$	$\mathbf{T_u}$ /2	$\mathbf{T_u}$ /3

Table 7 - Ziegler-Nichols method

3. MPU6050:

Figure 31 - MPU6050

3.1 Specifications:

- MPU-6050 module (3-axis gyroscope + 3-axis accelerometer)
- Chip: MPU-6050
- Input voltage: 3-5V
- Standard communication interface: I2C
- Chip 16bit AD converter, 16-bit data Output
- Gyro full-scale range: ± 250 500 1000 2000 °/s
- Accelerometer full-scale range: ± 2 ± 4 ± 8 ± 16g
- Standard jack: 2.54mm

3.2 Description:

MPU6050 is one of the first motion sensor in the world with up to 6 axes (expandable to 9-axis) sensors integrated in a single chip.

MPU-6050 uses MotionFusion proprietary technology of InvenSense that can run on mobile devices, controllers.

MPU-6050 integrated 6-axis sensors:

- 3-axis MEMS gyroscope.
- 3-axis MEMS accelerometer.

In addition, the MPU-6050 also has one unit dedicated hardware acceleration signal processing (Digital Motion Processor - DSP) collected by the sensor and perform the necessary calculations. This helps to significantly reduce calculation processing portion of the microcontroller, improve processing speed and for a faster response. This is one significant difference compared between the MPU-6050 with accelerometer and gyro sensors others.

MPU6050 can be combined with a magnetic field sensor (outside) to form the sensor 9 full angle via I2C interface.

3.3 Estimating the inclination angle with an Accelerometer:

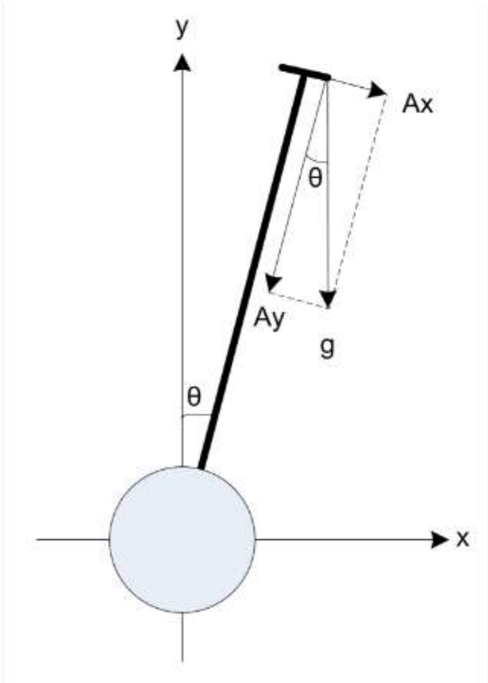


Figure 32 - Inclination angle calculation

Suppose that the robot is falling as illustrated above. The inclination angle can be calculated as:

$$\theta = tan^{-1}\frac{A_x}{A_y} = sin^{-1}\frac{A_x}{\sqrt{A_x^2 + A_y^2}} = sin^{-1}\frac{A_x}{g}$$

In the equation above, *Ax* is the *x* axis, *Ay* is the *y* axis that are the accelerometer reading on its *x* axis & *y* axis. When the robot is stationary, g is the gravitation constant.

In this case, because we are only interested in calculations where the inclination angle is small since our goal is to ensure that the deviation from balance is as small as possible, so further simply the equation:

$$\theta \approx sin(\theta) = \frac{A_x}{g} \quad \ldots (1)$$

By only measuring the x axis reading, we can get a raw estimate of the inclination angle in case assumption that the robot is standing still. In fact, when the robot isn't in balance it will accelerate towards the direction it's falling and thus the *x* axis reading will be slight more than *Ax* due to the acceleration. At the same time, the *y* axis reading will be slight less than *Ay*. As a result, the combined vector will deviate from *g*. But when the accelerometer is placed near the center of gravity of the robot, the acceleration along the x axis is small in near-balance condition. So the above equation will exhibit some small error, but it remains a good approximation of the inclination angle.

3.4 Estimating the inclination angle with a Gyroscope:

Gyroscope can measure the rate at which the rotation is taking place. And the rotation angle for a given time interval is governed by:

$$\theta(t) = \int_{t_1}^{t_2} G(t)dt$$

Where *G(t)* is the gyroscope reading with respect to the rotation direction. When the time interval is small enough, the gyroscope reading can be treated as a constant and thus the above equation can be approximated as:

$$\theta(t) \approx \theta(t_1) + G(t)(t_2 - t_1) = \theta(t_1) + G(t)\Delta t \quad \ldots (2)$$

Unlike the accelerometer, gyroscope measurement is largely immune to none angular movement and thus far less susceptible to vibrations and lateral accelerations mentioned previously. But since the angular measurement is cumulative, any minute error in measurements will manifest over time which causes the estimated angle to deviate from the true value. This is the so called drifting effect. Thus gyroscope alone cannot be used to reliably measure the inclination angle either.

3.5 Sensor fusion:

To address the issue of measurement noises and the limitations of measurements by either the accelerometer or gyroscope alone, we will need to combine the readings from both the accelerometer and the gyroscope in a meaningful way so that we could use the strengths from both sensors to obtain a more accurate result than either measurement alone could provide. This is the classic application of sensor fusion. Since accelerometers can provide accurate angle calculations when there is no acceleration and gyroscope can provide accurate short-time angle measurements? These complementary traits are ideal candidates for sensor fusion.

Like many processes in the physical world, sensor readings from the accelerometer and the gyroscope can be modelled as the true measurements with added white Gaussian noise (AWGN). Many filters in the least mean square filters family are suitable for this kind of stochastic process.

3.5.1 Kalman filter:

Kalman filter is one such adaptive filters that can be used to filter the sensor data from accelerometer and gyroscope. Even though the implementation of Kalman filter is quite straight forward, in order for the filter to be optimal we needed to know the precise underlying model of the system and we also need to be able to reliably estimate the noise covariance matrices. Without any reliable information on these parameters, the filter may still work but will not be in an optimal sense. Simpler methods can be used in situations where these parameters are unknown and can still achieve reasonably good result.

So, to keep the implementation simple, we used the method illustrated below where the estimated value is a linear combination of the filtered measurements from both the accelerometer and the gyroscope. Each sensor reading is multiplied by a fixed gain:

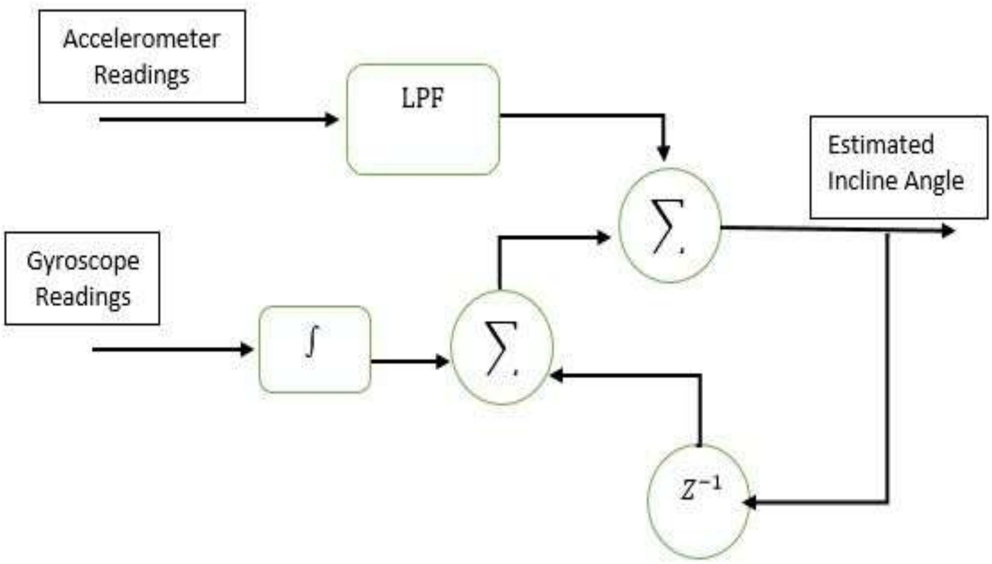


Figure 33 - Accelerometer and Gyroscope diagram

From the diagram above, we can see that in order to make the accelerometer readings more reliable, the readings are passed through a low pass filter (e.g. averaging over time) to smooth out any sudden change in values. And the gyroscope readings are integrated and then added to the previous estimate to give the current inclination angle reading. Each of the component is weighted and then added together to give the final estimate. Mathematically, the estimated angle can be expressed as follows:

$$\begin{cases} \theta_{est}\big|_{t_2} = \alpha(\theta_{est}\big|_{t_1} + G\Delta t) + \beta\theta_{A_x} & \dots (3) \\ \alpha + \beta = 1 & \dots (4) \end{cases}$$

Figure 34 - Calculation formula

Where G is the gyroscope reading and angle Ax is the angle calculated from the accelerometer reading in equation (1) above.

Figure 35 - Kalman Filter Data

But Kalman filter has some disadvantages for self-balancing system:

- Very complex to understand.
- Very hard, if not impossible, to implement on certain hardware (8-bit microcontroller etc.)
- Needs to calculate the coefficients of the matrices, the process-based error, measurement error, etc. that are not trivial.

3.5.2 Complementary filter:

In fact, Complementary filter manage both high-pass and low-pass filters simultaneously. The low pass filter filters high frequency signals (such as the accelerometer in the case of vibration) and low pass filters that filter low frequency signals (such as the drift of the gyroscope).In other words, on the short term, we use the data from the gyroscope, because it is very precise and not susceptible to external forces. On the long term, we use the data from the accelerometer, as it does not drift. By combining these filters, we get a good signal, without the complications of the Kalman filter In it's most simple form, the filter looks as follows:

$$angle = 0.98 * (angle + gyrData * dt) + 0.02 * (accData)$$

The gyroscope data is integrated every timestep with the current angle value. After this it is combined with the low-pass data from the accelerometer (already processed with atan2). The constants (0.98 and 0.02) have to add up to 1 but can of course be changed to tune the filter properly.

The data result come from Complementary filter:

Figure 36 – Complementary Filter Data

As we see, the filter is very easy and light to implement making it perfect for embedded systems.

3.5.3 Digital Motion Processor:

Beside above filters, we also can use the embedded Digital Motion Processor (DMP) is located within the MPU-6050 to fusion data from accelerometers, gyroscopes and processes them. The resulting data can be read from the DMP's registers, or can be buffered in a FIFO (a buffer inside MPU - 6050). The DMP has access to one of the MPU's external pins, which can be used for generating interrupts.

Figure 37 – DMP Data

The purpose of the DMP is to offload both timing requirements and processing power from the host processor. The DMP can be used as a tool in order to minimize power, simplify timing, simplify the software architecture, and save valuable MIPS on the host processor for use in the application.

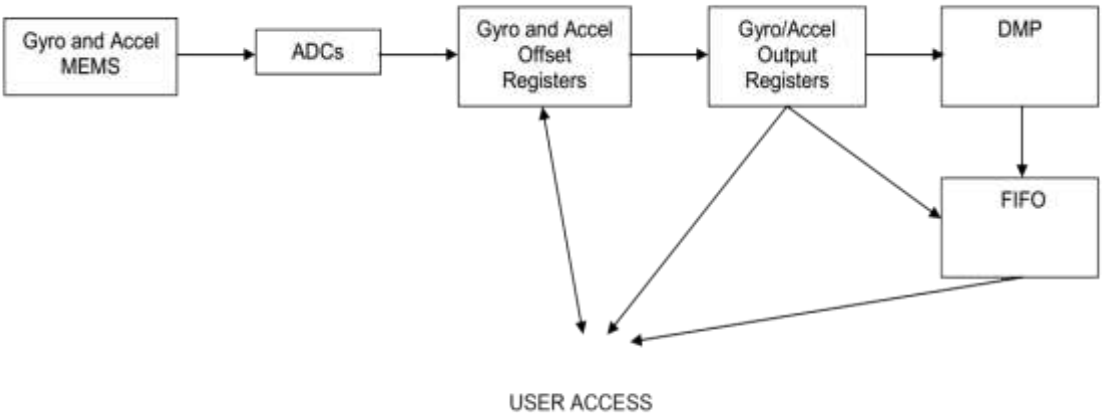


Figure 38 - Data Signal Diagram

3.5.4 Finding out the offsets:

Offsets are caused by a number of factors such as mechanical assembly, mounting, package damage and temperature fluctuations. So we need to work out what the offsets of our chip are and then remove them before processing the data. This can be very significant for many applications. The offset error alone can affect a tilt reading on a flat surface by as much as 12 degrees.

Let assign data from accelerometer, gyroscope Ax, Ay, Az, Gx, Gy, Gz respectively. When we place MPU6050 in horizontal position so our data must be zero with Ax, Ay, Gx, Gy, Gz and the remain Az equal 1G (16384 for default sensitivity). After that we measure the average acceleration and gyroscope over 10-20 samples and anything over 0G in the Ax, Ay, Gx, Gy, Gz and 1G in Az will be our offsets.

3.5.5 Conclusion:

We can see the data comes from DMP is little better than Kalman filter's data and Complementary filter's data. So we decide to use DMP to be main data fusion in this project.

3.6 Communication:

In this system, we have a lot of modules so we need to help them to communicate with each other.

Figure 39 - Modules and Protocols

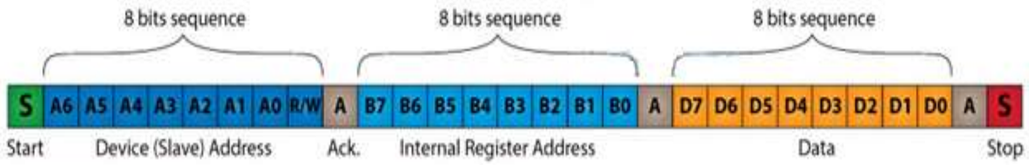
3.6.1 I2C:

I2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems.

I2C uses only two wires: SCL (serial clock) and SDA (serial data). Both need to be pulled up with a resistor to +Vdd. There are also I2C level shifters which can be used to connect to two I2C buses with different voltages.

Basic I2C communication is using transfers of 8 bits or bytes. Each I2C slave device has a 7-bit address that needs to be unique on the bus. Some devices have fixed I2C address while others have few address lines which determine lower bits of the I2C address. This makes it very easy to have all I2C devices on the bus with unique I2C address. There are also devices which have 10-bit address as allowed by the specification.

7-bit address represents bits 7 to 1 while bit 0 is used to signal reading from or writing to the device. If bit 0 (in the address byte) is set to 1 then the master device will read from the slave I2C device.



In normal state both lines (SCL and SDA) are high. The communication is initiated by the master device. It generates the Start condition (S) followed by the address of the slave device (B1). If the bit 0 of the address byte was set to 0 the master device will write to the slave device (B2). Otherwise, the next byte will be read from the slave device. Once all bytes are read or written (Bn) the master device generates Stop condition (P). This signals to other devices on the bus that the communication has ended and another device may use the bus.

Most I2C devices support repeated start condition. This means that before the communication ends with a stop condition, master device can repeat start condition with address byte and change the mode from writing to reading.

3.6.2 UART and Bluetooth:

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. UART uses two wire to transfer data : RX, TX.

Bluetooth is a wireless technology standard for exchanging data over short distances. It’s available on smartphone, so we use it to communicate with Android application.

4. Arduino Mega 2560:

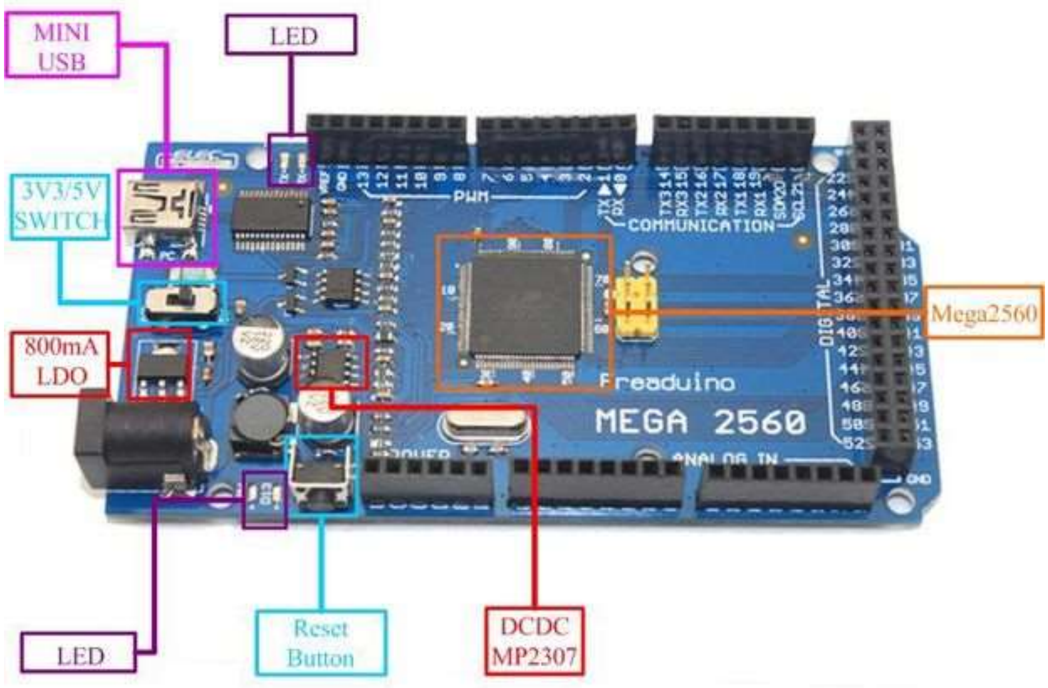


Figure 40 - Arduino Mega 2560

Technical specs:

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm
Weight	37 g

The Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an ACto-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

5. Module Control Engine L298:

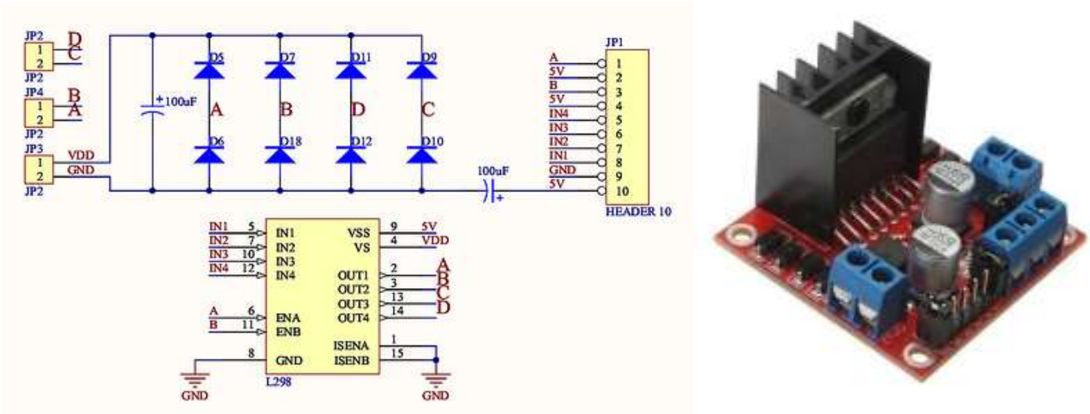


Figure 41 - L298

5.1 Specifications:

- Driver: L298N integrated dual H-bridges.
- Power supply: +5 V ~ +35 V
- Peak output current per channel: 2A
- Logic power output Vss: +5 V ~ +7 V
- Logic current: 0 ~ 36ma
- Max power: 20W(Temperature 75 Cesus)
- Working temperature: -25 ℃ ~ +130 ℃

5.2 Description:

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

Summary function pins of the L298:

- 4 pins INPUT: IN1, IN2, IN3, IN4 connected in turn with pins: 5, 7, 10, and 12 of L298. This is the control signal pins.
- 4 pins OUTPUT: OUT1, OUT2, OUT3, OUT4 (correspond with INPUT pins) connected in turn with pins: 2, 3, 13, and 14 of L298. These pins are connected to the motor.
- ENA and ENB pins used to control the L298 H-bridge circuit. If at logic "1" (powered with 5V) allowing H-bridge circuit operation, if at logic "0", the H-bridge circuit is not working

With our math above, you just need to pay attention to how to control the direction of rotation with L298:

- ENA = 0: Motor doesn't rotate with all inputs.
- ENA = 1:
  - INT1 = 1; INT2 = 0: Motor moves forward.
  - INT1 = 0; INT2 = 1: Motor moves reverse.
  - INT1 = INT2: Fast Motor stops
- ENB similar with INT3, INT4

6. Module Control Engine 43A BTS7960:

Figure 42 - Module Control Engine 43A BTS7960

This driver uses Infineon chips BTS7960 composed of high-power drive full H-bridge driver module with thermal over-current protection. Double BTS7960 H-bridge driver circuit, with a strong drive and braking, effectively isolating the microcontroller and motor driver! High-current 43A.

6.1 Specification:

- Double BTS7960 large current (43 A) H bridge driver.
- 5V isolate with MCU, and effectively protect MCU.
- 5V power indicator on board.
- Voltage indication of motor driver output end.
- Can solder heat sink.
- Just need four lines from MCU to driver module (GND. 5V. PWM1. PWM2).
- Isolation chip 5 V power supply (can share with MCU 5 V).
- Able to reverse the motor forward, two PWM input frequency up to 25kHZ.
- Two heat flow passing through an error signal output.
- Isolated chip 5V power supply (can be shared with the MCU 5V), can also use the on-board 5V supply.
- The supply voltage: 5.5V to 27V.

6.2 Pinout:

- VCC: Power Control (5V - 3V3)
- GND: Pin negative.
- R_EN = 0: Disable half H-bridge must.
- R_EN = 1: Enable half H-bridge must.
- L_EN = 0: Disable the left half of the H-bridge.
- L_EN = 1: Enable half H-bridge left.
- RPWM and LPWM: pin reverse controls and engine speed.
- RPWM = 1 and LPWM = 0: Motor forward rotation.
- LPWM RPWM = 0 and = 1: Motor reverse running
- RPWM = 1 and LPWM = 1 or RPWM = 0 and LPWM = 0: Stop.
- R_IS and L_IS: combined with resistor to limit the current through the H-bridge

7. Module Bluetooth HC06:

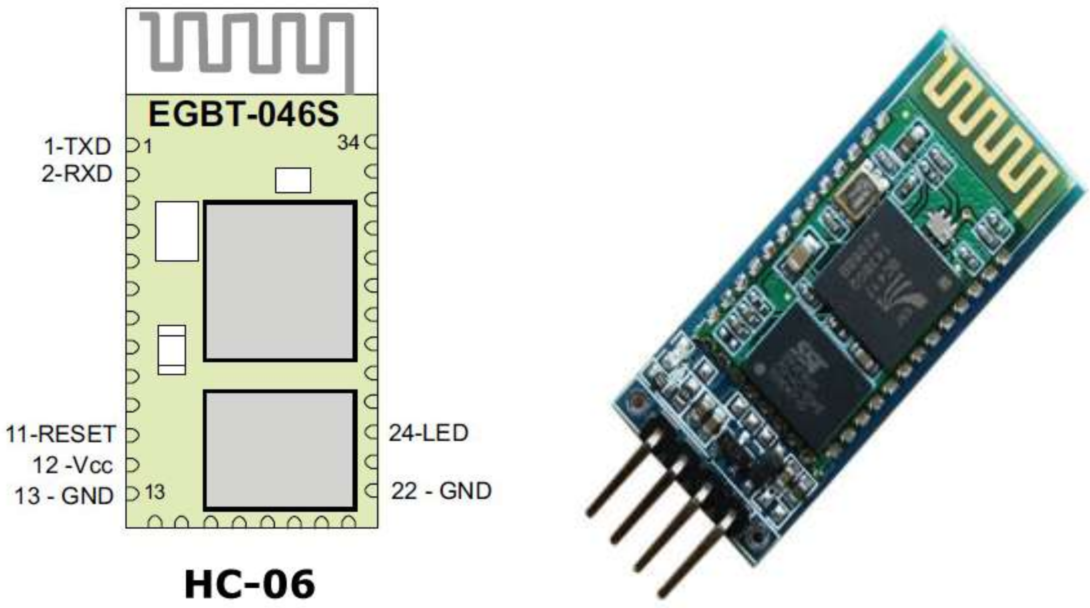


Figure 43 – Module Bluetooth HC06

SLAVE Bluetooth module allows microcontroller connected to the peripheral device: smartphone, laptop, Bluetooth usb ... sending and receiving two-way signals through Serial communication.

Bluetooth module integrated on the board allows you to use from 3.5V to 6V power supply board without worrying about voltage difference 3V - 5V may damage the board.

Bluetooth module includes in the order of 6 pins: KEY, VCC, GND, TX, RX, STATE

This is the Bluetooth module SLAVE means you cannot actively connected with the microcontroller, which need to use smartphone, laptop, Bluetooth usb... to detect a signal and connect (pair) from a smartphone, laptop, Bluetooth usb... After pair successfully, you can send and receive signals from the microcontroller to the devices.

8. Gear motors 12V 116RPM GH-1632T:

Figure 44 - Gear motors 12V 116RPM GH-1632T

Specifications:

Output: 2W

Voltage: 12V

Field No Download: 70mA

Speed: 116RPM 12V

Speed over voltage: 5-> 15V speed 48-> 145RPM

Weight: 120 g

9. Physical theory:

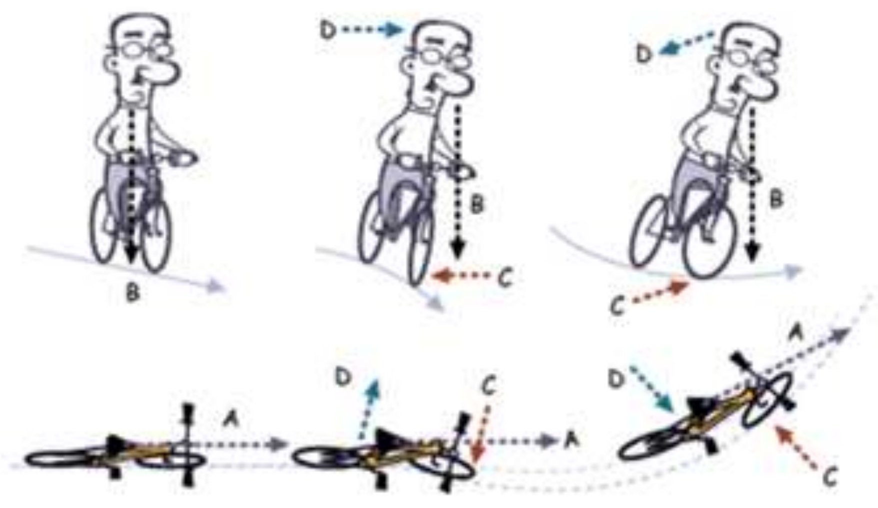


Figure 45 - Principles of operation of bicycles

A - Direction that the vehicle is straight forward.
B - Gravity.
C - Centripetal force.
D - Centrifugal force.

Handling a vehicle in equilibrium based on the principle as follows:

When the vehicle is inclined against the equilibrium position, it should have a force large enough to pull the vehicle from the inclined angle back to its original state. In fact, there is a common method called gyroscopes. However, it is found that the method is relatively common and needs a quite bulky model; therefore, our group do not perform in this way.

According to the theory of centrifugal force and empirical observations, when the vehicle is inclined, if the steering wheel is inclined towards the vehicle's direction, it will generate a centrifugal force pulling the vehicle towards the opposite direction. Thus, the goal of our group is to use the control algorithm PID to adjust the vehicle's rotation angle so that the centrifugal force is adequate and accurate. The ultimate aim is to keep the vehicle balance.

V. Software Design Description (SDD):

1. Purpose:

This document describes a general view of system which includes:

- System architectural design which describes system's model, system's layers, layer's component and the function of each layer.
- The detailed description of components and the relationship between each component.
- Sequence diagram to know how to the system run.
- User Interface design which describes system's screens and how users interact to systems. Each screen includes functions, types of input/ output and event handling.

2. Architecture Overview:

Robot two wheels self-balancing:

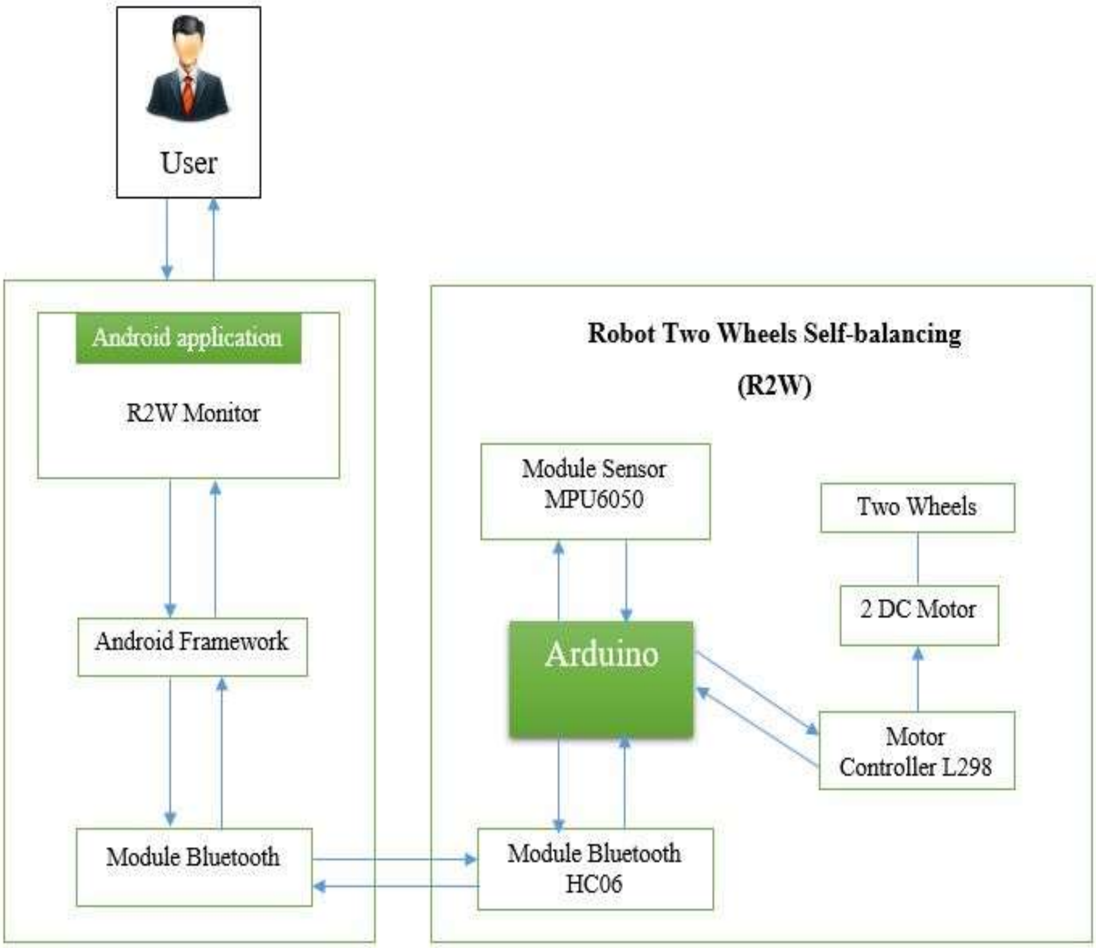


Figure 46 - System Architecture Diagram of R2W

Self-balancing Bike:

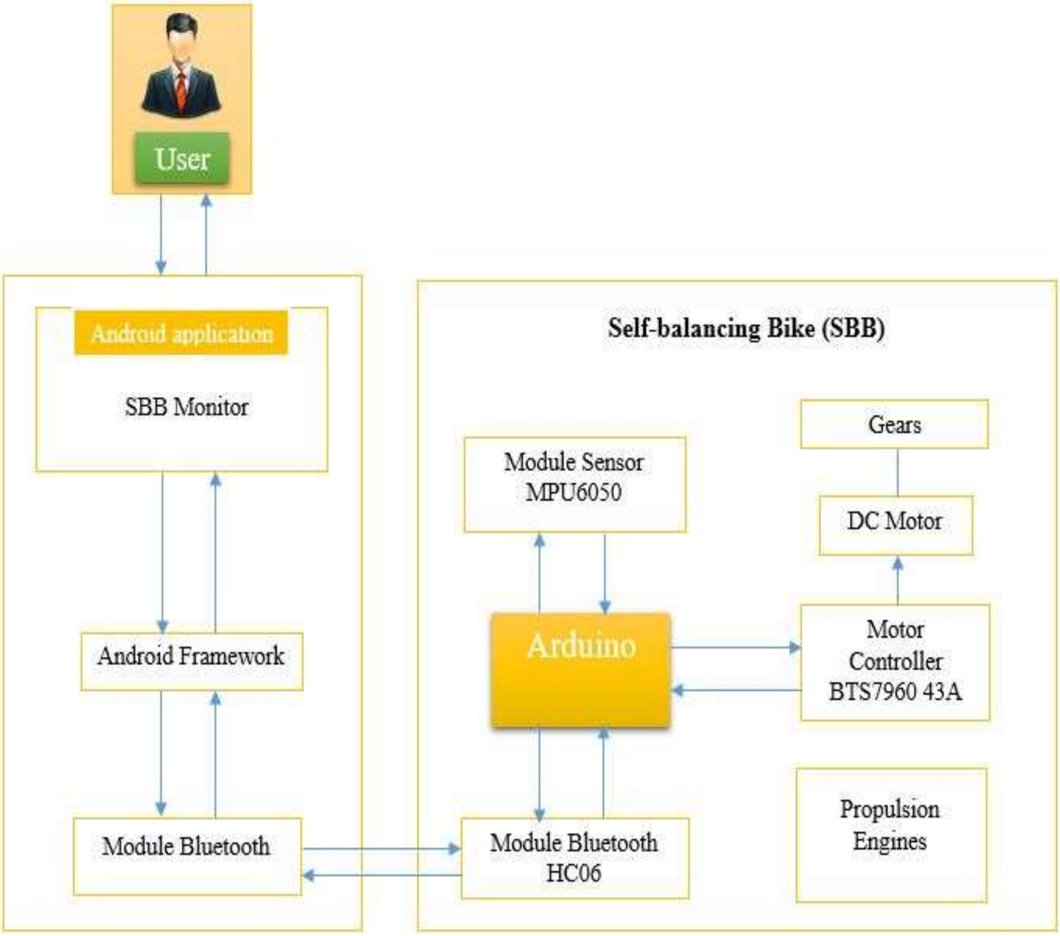


Figure 47 - System Architecture Diagram of SBB

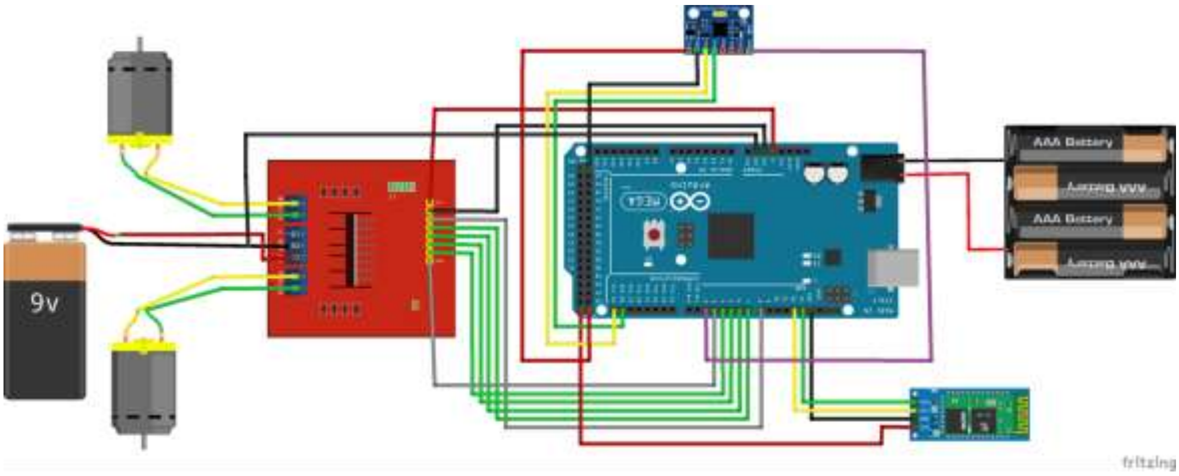


Figure 48 - Wire diagram

3. Class Diagram:

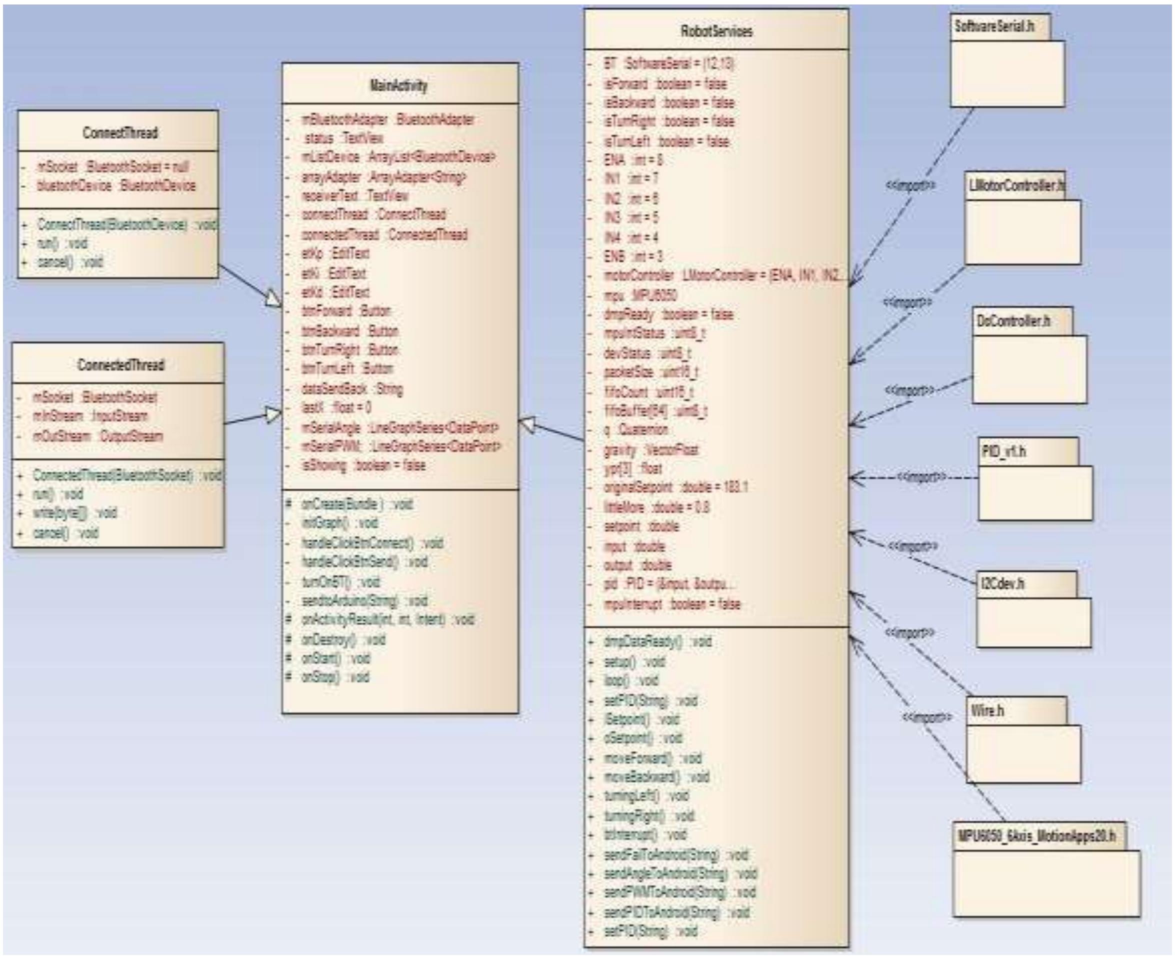


Figure 49 - Class Diagram

Main Activity			
Attributes			
Name	Type	Initialize	Description
mBluetoothAdapter	BluetoothAdapter		Represent the local device Bluetooth adapter.
status	TextView		The current status of Bluetooth connection.
mListDevice	ArrayList<BluetoothDevice>		List the device searched by Bluetooth

arrayAdapter	ArrayAdapter<String>		Adapter for dialog.
receiverText	TextView		Log data received from Arduino.
connectThread	ConnectThread		Background Thread to handle Bluetooth connecting.
connectedThread	ConnectedThread		Background Thread to handle Bluetooth connected.
etKp	EditText		Input value of $\mathbf{K_p}$
etKi	EditText		Input value of $\mathbf{K_i}$.
etKd	EditText		Input value of $\mathbf{K_d}$.
btnForward	Button		Forward button.
btnBackward	Button		Backward button.
btnTurnLeft	Button		Turn left button.
btnTurnRight	Button		Turn right button.
dataSendBack	String		Storing data that has sent.
mSerialAngle	LineGraphSeries<DataPoint>		Data for graph of angle.
mSerialPWM	LineGraphSeries<DataPoint		Data for graph of pulse.
isShowing	boolean	false	The displays status of the dialog.
btHandler	Handler		Receive data from Bluetooth and handling.
listener	View.onClickListener		Receive the event of the button click.
mBrbt	BroadcastReceiver		Receive the intent from Bluetooth module.
Methods			
Method Name	onCreate	Scope	Protected

Return	Return Type	Description	
N/A	None	Initialize basic variables.	
Parameter	Type	Default Value	
savedInstanceState	Bundle		
Method Name	initGraph	Scope	Private
Return	Return Type	Description	
Signal	Int	Initialize the graphs.	
Parameter	Type	Default Value	
Input	Double		
Method Name	handleClickBtnConnect	Scope	Private
Return	Return Type	Description	
N/A	None	Handling event click of the Connect button	
Parameter	Type	Default Value	
N/A	None		
Method Name	handleClickBtnSend	Scope	Private
Return	Return Type	Description	
N/A	None	Handling event click of the Send button	
Parameter	Type	Default Value	
N/A	None		
Method Name	turnOnBT	Scope	Private
Return	Return Type	Description	
N/A	None	Check current status of the device, if it off turn it on	
Parameter	Type	Default Value	
N/A	None		
Method Name	sendtoArduino	Scope	Private
Return	Return Type	Description	

N/A	None	Convert String into bytes to send to Arduino	
Parameter	Type	Default Value	
msg	String		
Method Name	onActivityResult	Scope	Protected
Return	Return Type	Description	
N/A	None	Receive results from enabled Bluetooth.	
Parameter	Type	Default Value	
requestCode	Int		
resultCode	Int		
data	Intent		
Method Name	onDestroy	Scope	Protected
Return	Return Type	Description	
N/A	None	Destroy Bluetooth connection.	
Parameter	Type	Default Value	
N/A	None		
Method Name	onStart	Scope	Protected
Return	Return Type	Description	
N/A	None	Attach Broadcast Receiver	
Para N/A meter	Type	Default Value	
N/A	None		
Method Name	onStop	Scope	Protected
Return	Return Type	Description	
N/A	None	No attach Broadcast Receiver	
Parameter	Type	Default Value	
N/A	None		

Connect Thread			
Attributes			
Name	Type	Initialize	Description
mSocket	BluetoothSocket	null	Socket of Bluetooth.
bluetoothDevice	BluetoothDevice		Bluetooth devices need to connect.
Methods			
Method Name	ConnectThread	Scope	Private
Return	Return Type	Description	
N/A	None	Initialize class.	
Parameter	Type	Default Value	
Device	BluetoothDevice		
Method Name	run	Scope	Public
Return	Return Type	Description	
N/A	None	Make a connection.	
Parameter	Type	Default Value	
N/A	None		
Method Name	cancel	Scope	Public
Return	Return Type	Description	
N/A	None	Make a disconnection.	
Parameter	Type	Default Value	
N/A	None		

Connected Thread			
Attributes			
Name	Type	Initialize	Description
mSocket	BluetoothSocket		Socket is already connected
mInStream	InputStream		Input Bluetooth stream data

mOutStream	OutputStream		Output Bluetooth stream data
Methods			
Method Name	ConnectedThread	Scope	Public
Return	Return Type	Description	
N/A	None	Initialize class.	
Parameter	Type	Default Value	
Socket	BluetoothSocket		
Method Name	run	Scope	Public
Return	Return Type	Description	
N/A	None	Receive data stream from Bluetooth	
Parameter	Type	Default Value	
N/A	None		
Method Name	write	Scope	Public
Return	Return Type	Description	
N/A	None	Transfer data to Bluetooth.	
Parameter	Type	Default Value	
bytes	byte[]		
Method Name	cancel	Scope	Public
Return	Return Type	Description	
N/A	None	Make a disconnection.	
Parameter	Type	Default Value	
N/A	None		

Robot Services			
Attributes			
Name	Type	Initialize	Description

BT	SoftwareSerial	(12,13)	Set port 12 be RX and port 13 be TX
isForward	boolean	false	Check robot is going straight or not.
isBackward	boolean	false	Check robot is going backwards or not
isTurningLeft	boolean	false	Check robot is turning left or not
isTurningRight	boolean	false	Check robot is turning right or not
isEnough	boolean		Check data that has sent from Android
ENA	Int	8	Set up pin Enable A for L298
IN1	Int	7	Set up pin Input 1 for L298
IN2	Int	6	Set up pin Input 2 for L298
ENB	Int	3	Set up pin Enable B for L298
IN3	Int	5	Set up pin Input 3 for L298
IN4	Int	4	Set up pin Input 4 for L298
motorController	LMotorController	(ENA, IN1, IN2, ENB, IN3, IN4, 1, 1)	Through L298 to control motor
mpu	MPU6050		
dmpReady	boolean	false	Set true if DMP initialization was successful
mpuIntStatus	uint8_t		Holds actual interrupt status byte from MPU
devStatus	uint8_t		Return status after each device operation (0 = success, !0 = error)
packetSize	uint8_t		Expected DMP packet size (default is 42 bytes)
fifoCount	uint8_t		Count of all bytes currently in FIFO
fifoBuffer[64]	uint8_t		FIFO storage buffer

q	Quaternion		[w, x, y, z] quaternion container
gravity	VectorFloat		[x, y, z] gravity vector
ypr[3]	Float		[yaw, pitch, roll] yaw/pitch/roll container and gravity vector
originalSetpoint	Double	183.1	Setting the balance position of the robot (the angle for the robot is self-balance).
littleMore	Double	0.8	The angle value for robot can go straight, backward, turn right or turn left.
setpoint	Double		The current angle for robot is self-balance.
input	Double		The tilt angle of the robot
output	Double		Pulse of L298 for motor control
pid	PID	(&input, &output, &setpoint, 75, 240, 4, DIRECT)	Class PID
mpuInterrupt	Boolean	False	Interrupt signal of the MPU6050
Methods			
Method Name	dmpDataReady	Scope	Void
Return	Return Type	Description	
N/A	None	Check availability of data from MPU6050	
Parameter	Type	Default Value	
N/A	None		
Method Name	setup	Scope	Void
Return	Return Type	Description	
N/A	None	Setting the basic parameters	
Parameter	Type	Default Value	
N/A	None		
Method Name	loop	Scope	Void

Return	Return Type	Description	
N/A	None	Execute the command loop	
Parameter	Type	Default Value	
N/A	None		
Method Name	btInterrupt	Scope	Void
Return	Return Type	Description	
N/A	None	Checking data from Bluetooth module	
Parameter	Type	Default Value	
N/A	None		
Method Name	iSetpoint	Scope	Void
Return	Return Type	Description	
N/A	None	Increase Setpoint	
Parameter	Type	Default Value	
N/A	None		
Method Name	dSetpoint	Scope	Void
Return	Return Type	Description	
N/A	None	Decrease Setpoint	
Parameter	Type	Default Value	
N/A	None		
Method Name	oSetpoint	Scope	void
Return	Return Type	Description	
N/A	None	Returns the original setpoint	
Parameter	Type	Default Value	
N/A	None		
Method Name	moveForward	Scope	void
Return	Return Type	Description	
N/A	None	Change the robot state to go straight	
Parameter	Type	Default Value	
N/A	None		
Method Name	moveBackward	Scope	Void
Return	Return Type	Description	

N/A	None	Change the robot state to go backward	
Parameter	Type	Default Value	
N/A	None		
Method Name	TurningLeft	Scope	void
Return	Return Type	Description	
N/A	None	Change the robot state to turn left	
Parameter	Type	Default Value	
N/A	None		
Method Name	TurningRight	Scope	void
Return	Return Type	Description	
N/A	None	Change the robot state to turn right	
Parameter	Type	Default Value	
N/A	None		
Method Name	SendFailToAndroid	Scope	void
Return	Return Type	Description	
N/A	None	Notify to smart phone the data error	
Parameter	Type	Default Value	
N/A	None		
Method Name	SendAngleToAndroid	Scope	void
Return	Return Type	Description	
N/A	None	Send inclination angle of the robot on the phone	
Parameter	Type	Default Value	
N/A	None		
Method Name	sendPWMToAndroid	Scope	Void
Return	Return Type	Description	
N/A	None	Send PWM (Pulse Width Modulation) value from the robot to the phone.	
Parameter	Type	Default Value	
N/A	None		
Method Name	sendPIDToAndroid	Scope	Void

Return	Return Type	Description	
N/A	None	Send current PID values of the robot to the phone	
Parameter	Type	Default Value	
N/A	None		
Method Name	setPID	Scope	Void
Return	Return Type	Description	
N/A	None	Function for setting direct PID values from Bluetooth data to Robot	
Parameter	Type	Default Value	
data	String		

4. Sequence Diagram:

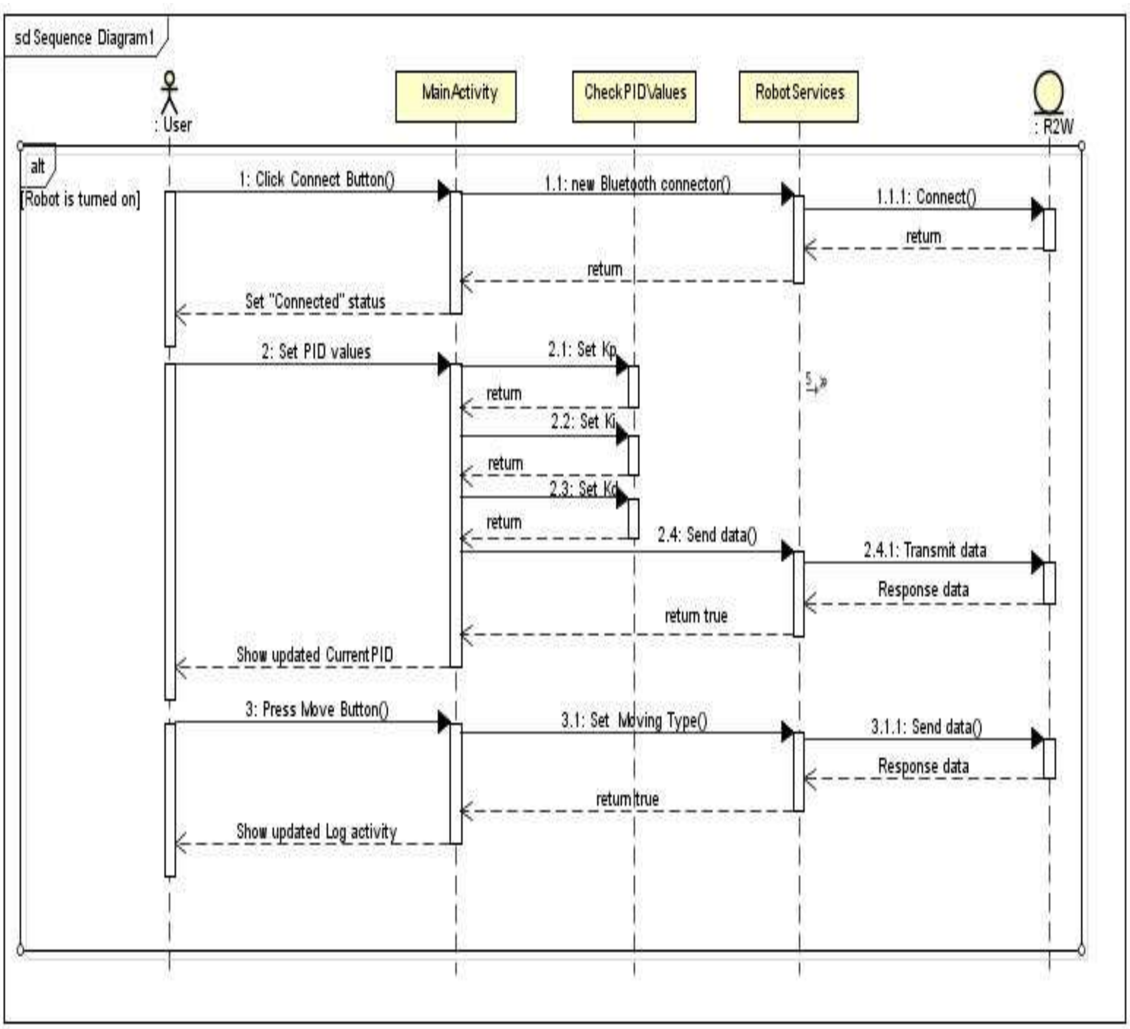


Figure 50 - Sequence Diagram of Robot Two Wheels Self-balancing

Connect Bluetooth:

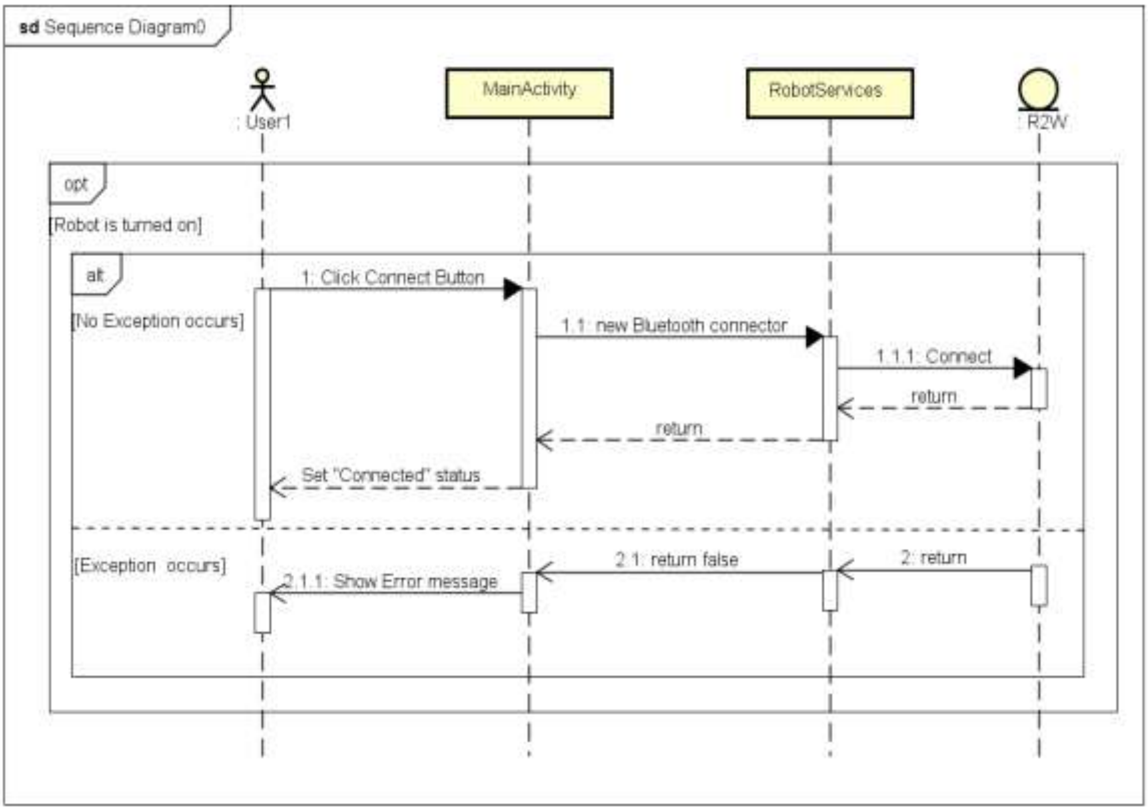


Figure 51 - Connect Bluetooth to Sequence Diagram

Control Movement:

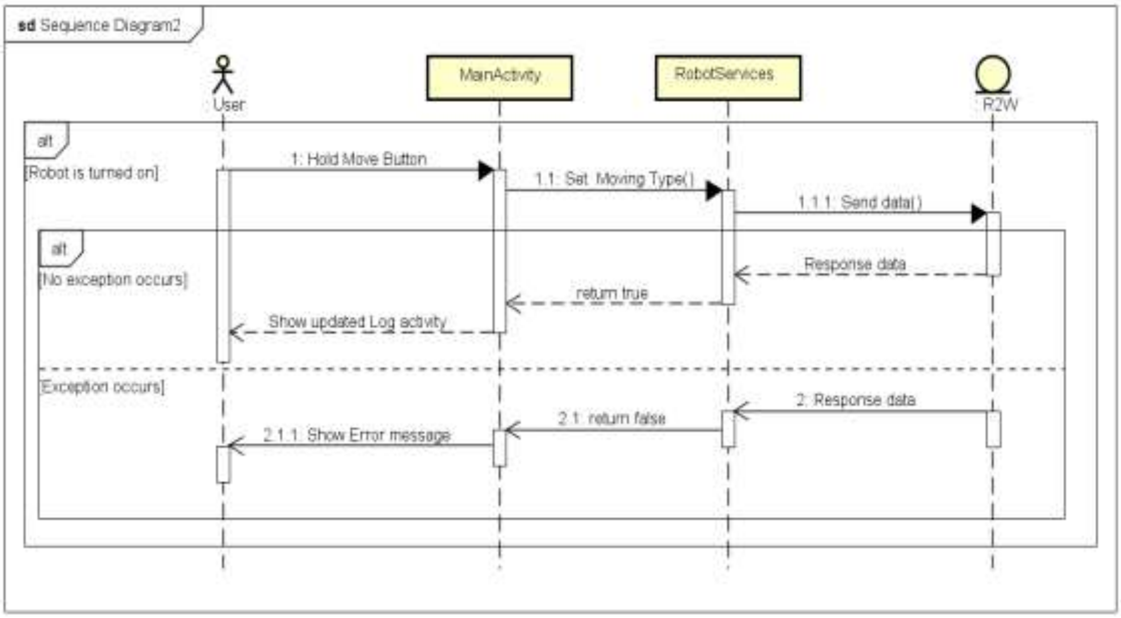


Figure 52 - Moving Controller of Sequence Diagram

Setting PID values:

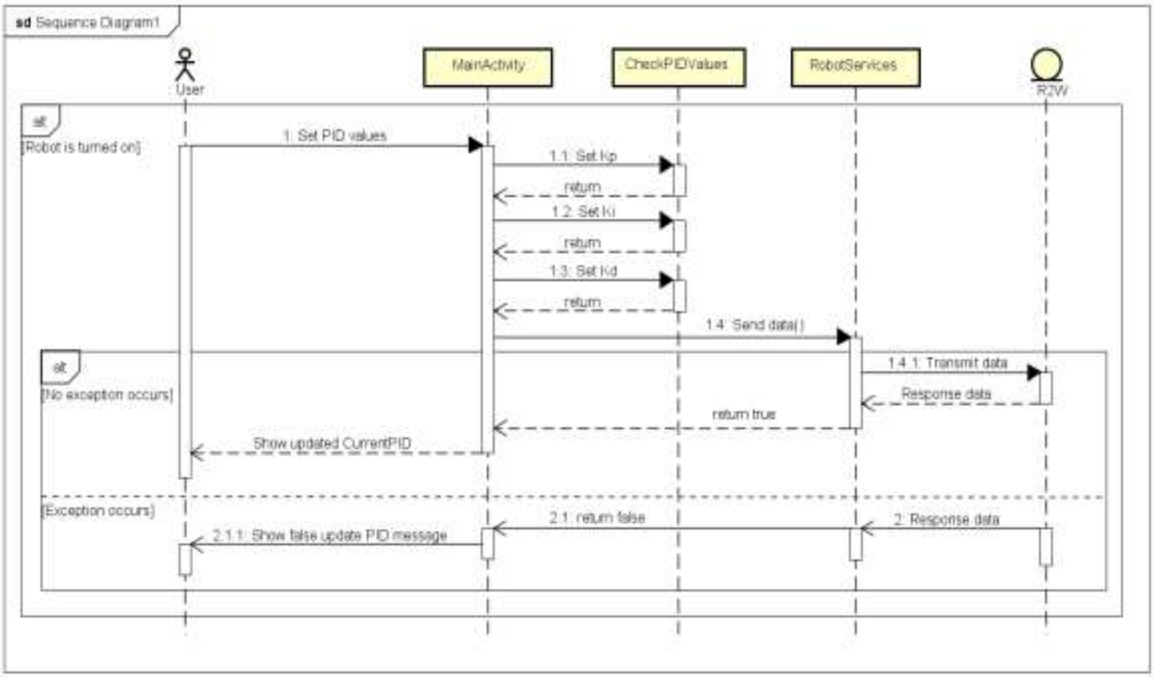


Figure 53 - Set PID values of Sequence Diagram

5. User Interface design/ Hardware interface design:

5.1 Android Application Interface:

Figure 54 - Android application interface

5.2 Robot Two Wheels Self-balance Image:

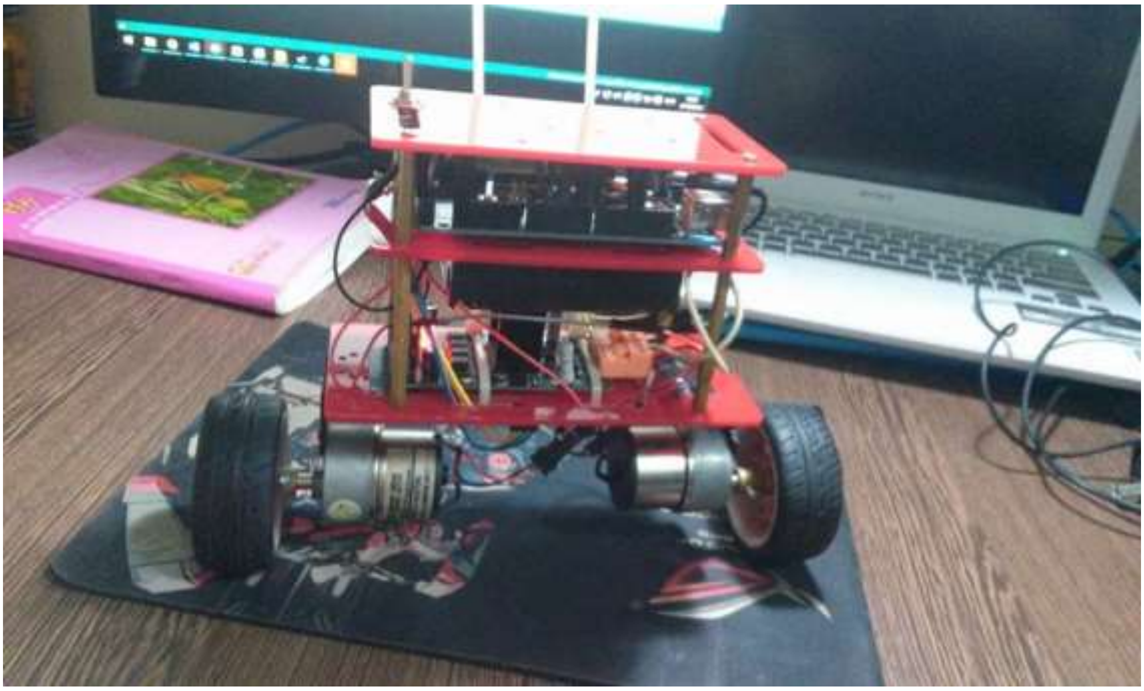


Figure 55 - The Robot two wheels self-balance

5.3 Self-balancing Bike Image:

Figure 56 - The Self-Balance Bike

VI. System Implementation and Testing:

1. Introduction:

1.1 Purpose:

The primary purpose of this report is to detect software failures so that defects may be discovered and correct to ensure that our project is thoroughly tested and resulting in a successful implementation project.

1.2 Test approach:

The purpose of this section is to verify and ensure that R2W's function meets its design specification and other requirements from user. The following part will describe which features to be tested.

ID	Test Stages	Description
1	**Unit Testing**	Unit testing will be done by the developer and approved by the development team leader.
2	**Integration testing**	Integration testing will be performed by testers. Requirements of the system will be tested in functional flow. Starting after unit testing complete for each flow. Focuses on specific areas of uses case when all requirement are completed, integration test should be performed to ensure all components incorporate well.
3	**System testing**	System Testing will be performed by the tester and development team leader with assistance from the individual developers as required. No specific test tools are available for this project. Programs will enter into System/Integration test after all critical defects have been correct.
4	**Acceptance testing**	Acceptance testing consist of Alpha Test and Beta Test will be executed by all team members, stand at end user point of view. Determine whether a system satisfies the requirement analysis phase. Finding defects is not the main focus in this stage. Acceptance testing will access the system's readiness for deployment and using.

1.3 Test environment:

Test tool: A Phone with Android Operating System and having Bluetooth module to connect to R2W.

1.4 Test plans:

No.	Component		Feature Name	Function to be tested
1	Electronic Components	Arduino Mega 2560	Operability	Checking components that can operate following their characteristics in datasheet.
		MPU6050		
		L298		
		HC06		
		Motor GH-1632T		
2	User's Interface		Screen Display	Checking the display of element.
3	Main Function		Connection	Testing connect between mobile phone and robot or bike via Bluetooth.
			Setting values	Testing operability of the robot or bike with PID values are transmitted from mobile phone.
			User's interaction	Testing the self-balance and movement of robot or bike by the control of the user.

2. Test Case:

2.1 Robot Two Wheels Self-Balancing:

Module Code	*Electronic Components*			
Test requirement	*Testing all electronic components of the project*			
Tester				
Pass	Fail	Untested	N/A	Number of Test cases
5	0	0	0	5

ID	Test Case Description	Test Case Procedure	Expected Output	Inter-test case Dependence	Result	Test date	Note
[Electronic Components-]	Arduino Mega 2560	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/21/2016	
[Electronic Components-1]	MPU6050	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/21/2016	
[Electronic Components-2]	L298	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/21/2016	
[Electronic Components-3]	HC06	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/21/2016	
[Electronic Components-4]	Motor	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/21/2016	

Cover | Test case List | Electronic Components | User's Interface | Main Function | Test Report

Figure 57 - Electronic Components Test Case of R2W

Module Code	*User's Interface*			
Test requirement	*Testing user's interface on mobile phone display*			
Tester				
Pass	Fail	Untested	N/A	Number of Test cases
4	0	0	0	4

ID	Test Case Description	Test Case Procedure	Expected Output	Inter-test case Dependence	Result	Test date	Note
[User's Interface-]	Display text box to input Kp, Kd, Ki values	Display on Main Screen	interface displays information Extension of student tuition		Pass	7/21/2016	
[User's Interface-1]	Display 4 buttons to control movement of Robot	Display on Main Screen	interface displays information Register to retake subject		Pass	7/21/2016	
[User's Interface-2]	Display graph of PID values and angle from MPU6050	Display on Main Screen	interface displays information Register for grade point improvement		Pass	7/21/2016	
[User's Interface-3]	Display Connect Status	Display on Main Screen	1.Display "Connected" when Robot connected to Mobile Phone by Bluetooth. 2. Display "Disconnected" when Robot did not connect to Mobile Phone.		Pass	7/21/2016	

Cover | Test case List | Electronic Components | User's Interface | Main Function | Test Report

Figure 58 - User's Interface Test Case of R2W

Code	Main Function
Test requiremen	Show the class action function
Tester	

Pass	Fail	Untested	N/A	Number of Test cases
7	0	0	0	7

ID	Test Case Description	Test Case Procedure	Expected Output	Inter-test case Dependence	Resu	Test date	Note
[Main Function-]	Connection	Connect from Android Smart Phone to Robot by Bluetooth.	1. Light of Bluetooth is on. 2. On phone, show		Pass	7/21/2016	
[Main Function-1]	Set PID values	1. Set Kp textbox equals 0. 2. Set Ki textbox equals 0.	Robot falls immediately.		Pass	7/21/2016	
[Main Function-2]	Robot stands on flat surface.	Robot stand self-balancing.	1. Robot stand on flat surface and self-balancing with falling in 30 minutes.		Pass	7/21/2016	
[Main Function-3]	Robot go forward	1. In the main view, user hold center button forward. 2. After 5 seconds, user release the center button.	1. Robot move forward without fall. 2. After 5 seconds, robot stop moving and keep balancing.		Pass	7/21/2016	
[Main Function-4]	Robot go backward	1. In the main view, user hold center button backward. 2. After 5 seconds, user release the center button.	1. Robot move backward without fall. 2. After 5 seconds, robot stop moving and keep balancing.		Pass	7/21/2016	
[Main Function-5]	Robot turn right	1. In the main view, user hold center button to the right. 2. After 5 seconds, user release the center button.	1. Robot turn right without fall. 2. After 5 seconds, robot stop turning and keep balancing.		Pass	7/21/2016	
[Main Function-6]	Robot turn left	1. In the main view, user hold center button to the left. 2. After 5 seconds, user release the center button.	1. Robot turn left without fall. 2. After 5 seconds, robot stop turning and keep balancing.		Pass	7/21/2016	

Cover | Test case List | Electronic Components | User's Interface | Main Function | Test Report

Figure 59 - Main Function Test Case of R2W

2.2 Self-balancing Bike:

Module Code	Electronic Components
Test requirement	Testing all electronic components of the project
Tester	

Pass	Fail	Untested	N/A	Number of Test cases
5	0	0	0	5

ID	Test Case Description	Test Case Procedure	Expected Output	Inter-test case Dependence	Result	Test date	Note
[Electronic Components-]	Arduino Mega 2560	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/22/2016	
[Electronic Components-1]	MPU6050	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/22/2016	
[Electronic Components-2]	Module Control BTS7960	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/22/2016	
[Electronic Components-3]	HC06	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/22/2016	
[Electronic Components-4]	Motor	Checking components that can operate following their characteristics in datasheet.	Works well and no error occurred		Pass	7/22/2016	

Figure 60 - Electronic Components Test Case of SSB

Module Code	*User's Interface*			
Test requirement	*Testing user's interface on mobile phone display*			
Tester				
Pass	**Fail**	**Untested**	**N/A**	**Number of Test cases**
4	0	0	0	4

ID	Test Case Description	Test Case Procedure	Expected Output	Inter-test case Dependence	Result	Test date	Note
[User's Interface-]	Display text box to input Kp, Kd, Ki values	Display on Main Screen	Full display		Pass	7/22/2016	
[User's Interface-1]	Display 4 buttons to control movement of Robot	Display on Main Screen	Full display		Pass	7/22/2016	
[User's Interface-2]	Display graph of PID values and angle from MPU	Display on Main Screen	Full display		Pass	7/22/2016	
[User's Interface-3]	Display Connect Status	Display on Main Screen	1.Display "Connected" when Bike connected to Mobile Phone by Bluetooth. 2. Display "Disconnected" when Bike did not connect to Mobile Phone.		Pass	7/22/2016	

Cover | Test case List | Electronic Components | User's Interface | Main Function | Test Report

Figure 61 - User's Interface Test Case of SSB

Code	*Main Function*			
Test requirement	*Show the class action function*			
Tester				
Pass	**Fail**	**Untested**	**N/A**	**Number of Test cases**
2	3	0	0	5

ID	Test Case Description	Test Case Procedure	Expected Output	Inter-test case Dependence	Result	Test date	Note
[Main Function-]	Connection	Connect from Android Smart Phone to Robot by Bluetooth.	1. Light of Bluetooth is on. 2. On phone, show		Pass	7/22/2016	
[Main Function-1]	Set PID values	1. Set Kp textbox equals 0. 2. Set Ki textbox equals 0. 3. Set Kp textbox equals 0. 4. Click Send button.	Bike falls immediately.		Pass	7/22/2016	
[Main Function-2]	Bike self-balacing	1. User start motor.	Bike move forward without fall.		Fail	7/22/2016	
[Main Function-3]	Bike turn right	1. In the main view, user hold right. button. 2. After 5 seconds, user release the right button.	1. Bike turns right without fall. 2. After 5 seconds, Bike stop turning and keep forward and self-balancing.		Fail	7/22/2016	
[Main Function-4]	Bike turn left	1. In the main view, user hold left button. 2. After 5 seconds, user release the left button.	1. Bike turns left without fall. 2. After 5 seconds, Bike stop turning and keep forward and self-balancing.		Fail	7/22/2016	

Figure 62 - Main Functions Test Case of SSB

3. Test Report:

TEST REPORT

Project Name	*Robot two wheels self-balancing*	Creator		Luu Ngoc Viet Son
Project Code	*R2W*	Reviewer/ Approver		Tran Van Tuan
Document Code	*R2W_Test Report_vx.x*	Issue Date		7/21/2016
Notes				

No	Module code	Pass	Fail	Untested	N/A	Number of test cases
1	Electronic Components	5	0	0	0	5
2	User's Interface	4	0	0	0	4
3	Function in Android app	7	0	0	0	7
	Sub total	16	0	0	0	16

Test coverage	100.00 %
Test successful coverage	100.00 %

Figure 63 - Test Report of R2W

TEST REPORT

Project Name	*Robot two wheels self-balancing*	Creator		Luu Ngoc Viet Son
Project Code	*R2W*	Reviewer/ Approver		Tran Van Tuan
Document Code	*R2W_Test Report_vx.x*	Issue Date		7/22/2016
Notes				

No	Module code	Pass	Fail	Untested	N/A	Number of test cases
1	Electronic Components	5	0	0	0	5
2	User's Interface	4	0	0	0	4
3	Function in Android app	2	3	0	0	5
	Sub total	11	3	0	0	14

Test coverage	100.00 %
Test successful coverage	78.57 %

Figure 64 - Test Report of SSB

VII. System User's Manual (SUM):

1. Setting up hardware:

Supply power for robots

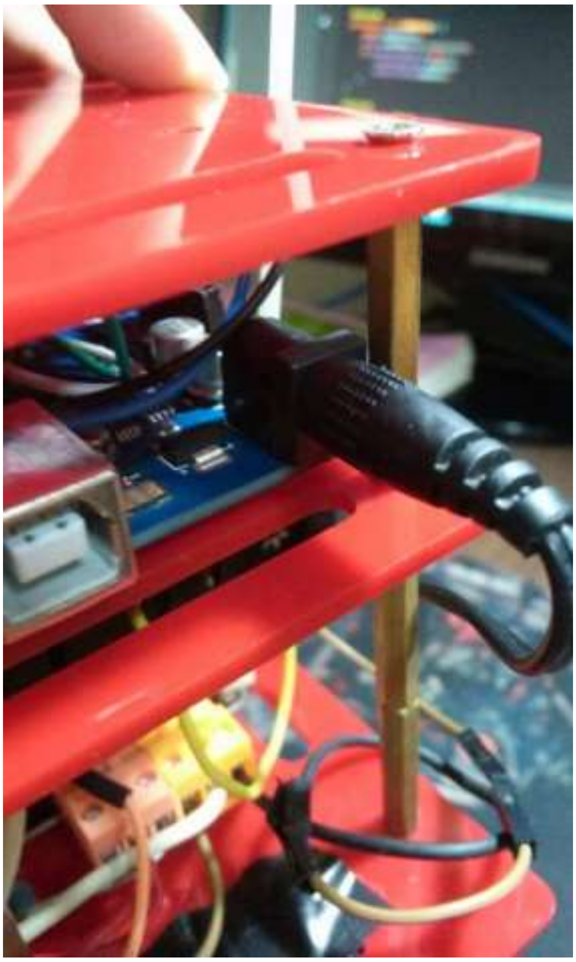


Figure 65 - Supply power for Robot

Start the robot

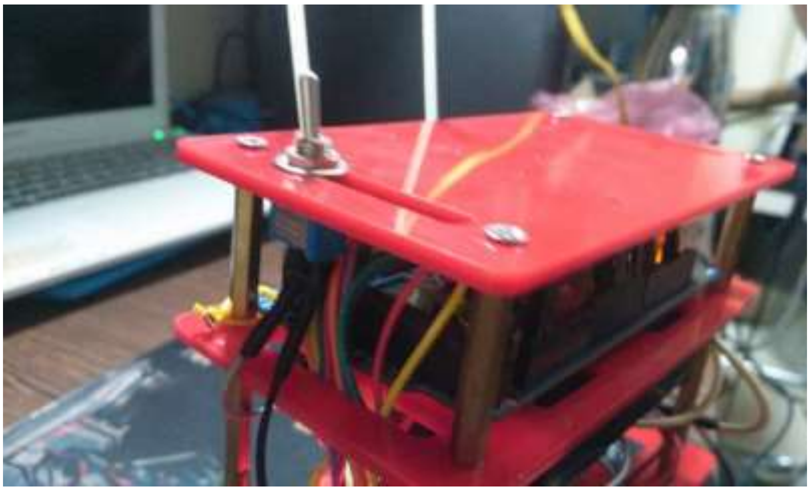


Figure 66 - Press the switch to start the Robot

2. Android application:

2.1 Connect Bluetooth:

- Choose connect Bluetooth on setting of smart phone
- Click on “Yes” to search the device that need to connect
- After connect successful, screen display “Connected” status

Figure 67 - Choose connect Bluetooth on the setting of smart phone

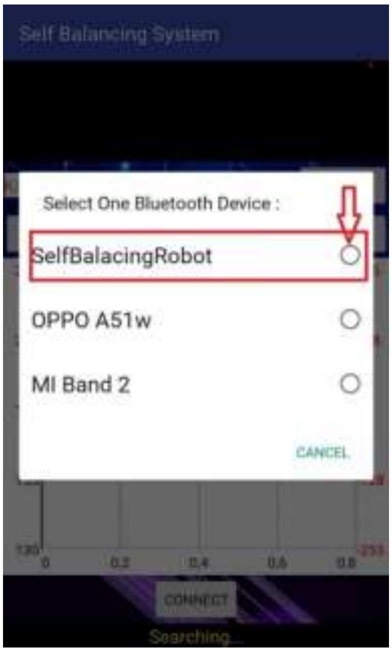


Figure 68 - Choose device that want to connect

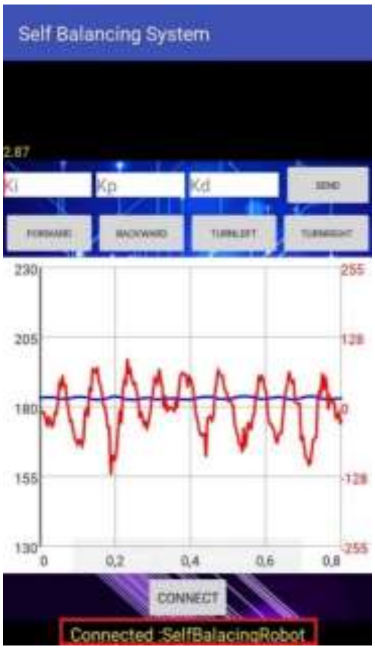


Figure 69 - Connected successful to Robot

2.2 Control movement:

Move forward:

- Click “FORWARD” button on the main screen.
- Robot will go straight and display “moveForward” status on the main screen.
- Click “Stop” button to stop the robot.

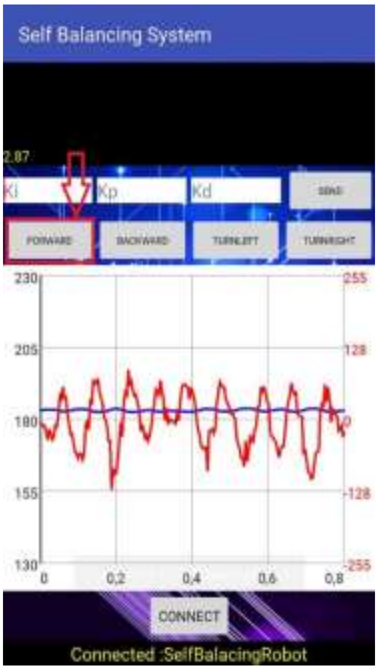


Figure 70 - Click on Forward button

Figure 71 - Show status

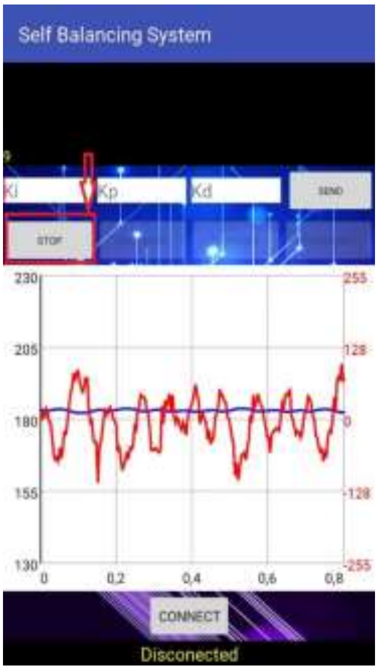


Figure 72- Click Stop button

Move backward:

- Click "BACKWARD" button on the main screen.
- Robot will go straight and display "moveBackward" status on the main screen.
- Click "Stop" button to stop the robot.

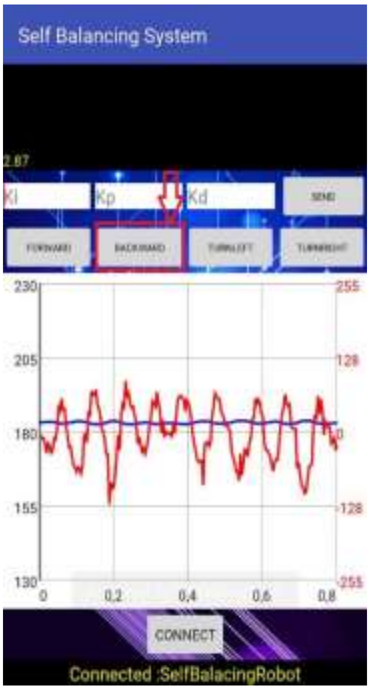


Figure 73 - Click "Backward" button

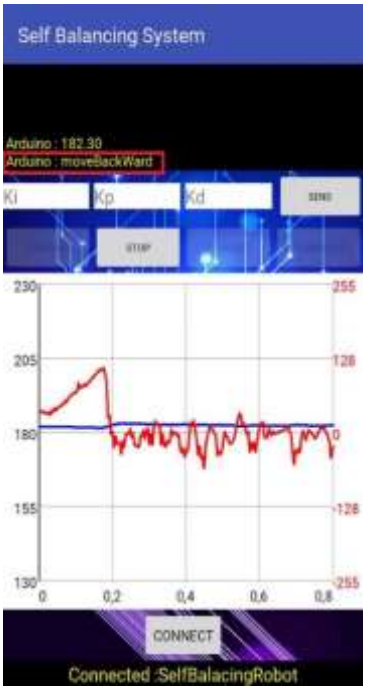


Figure 74 - Display status

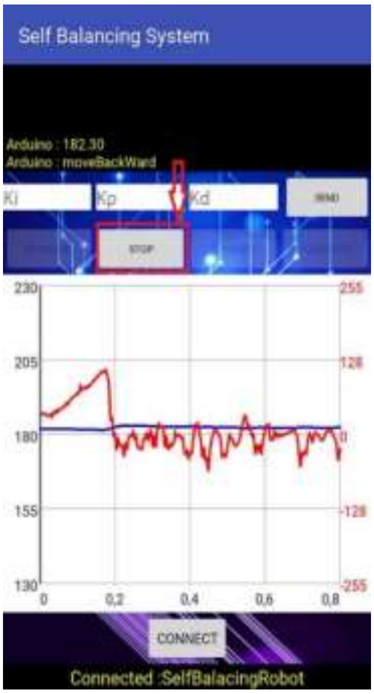


Figure 75 – Click "Stop" button

Turn left:

- Click “TURNLEFT” button on the main screen.
- Robot will go straight and display “Turning Left” status on the main screen.
- Click “Stop” button to stop the robot.

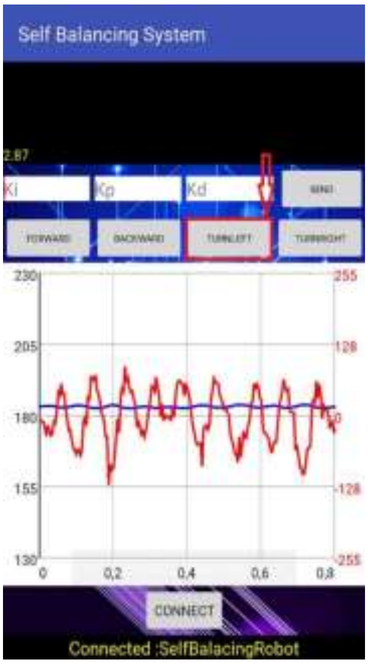


Figure 76 - Click Turn Left button

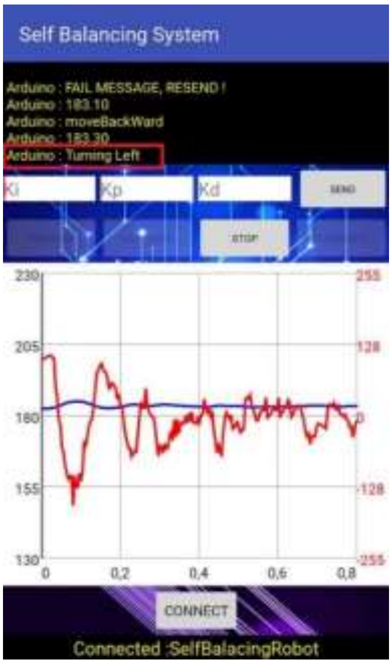


Figure 77 - Show status

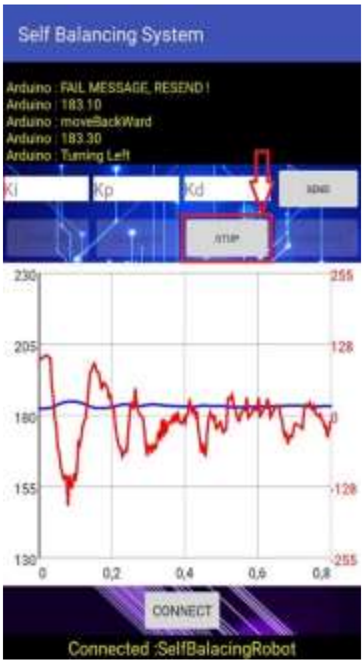


Figure 78- Click Stop button

Turn right:

- Click “TURNRIGHT” button on the main screen.
- Robot will go straight and display “Turning Right” status on the main screen.
- Click “Stop” button to stop the robot.

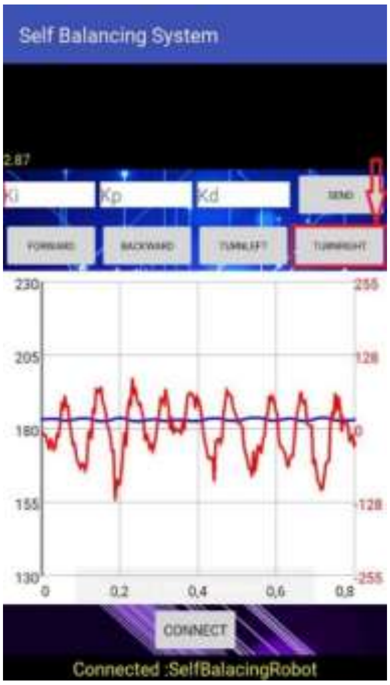


Figure 79 - Click Turn Right button

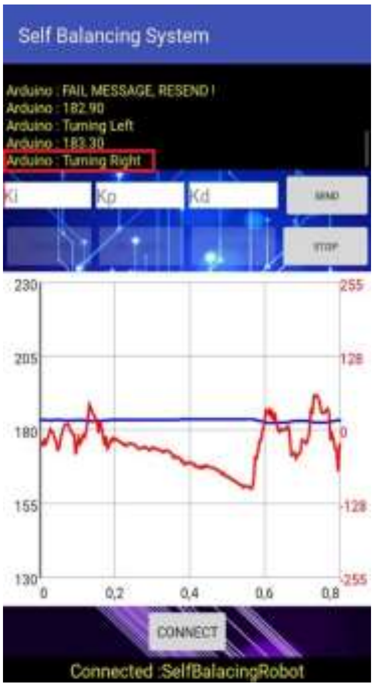


Figure 80 - Show status

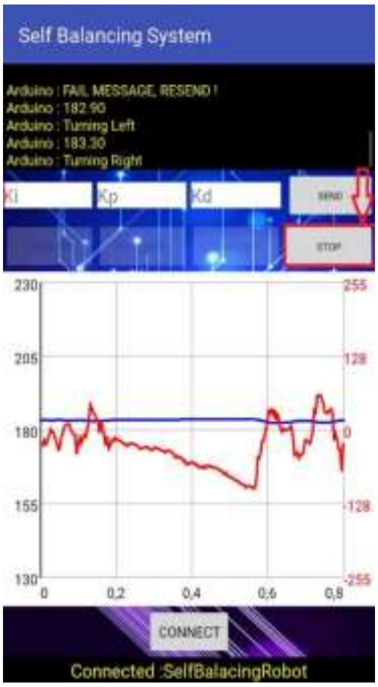


Figure 81 - Click Stop button

2.4 Setting PID values:

- Input new PID values.
- Display status on the main screen.

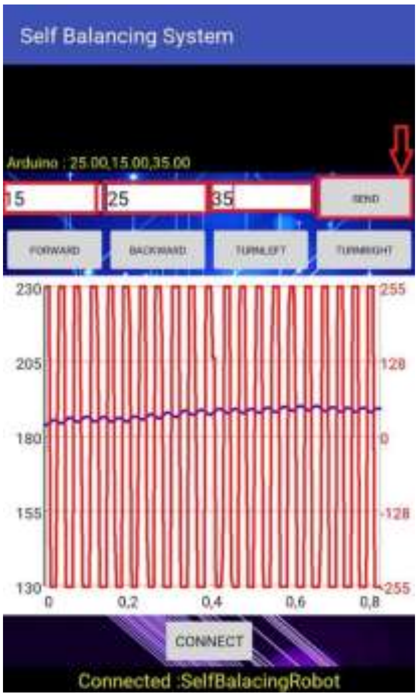


Figure 82 - Input new PID values

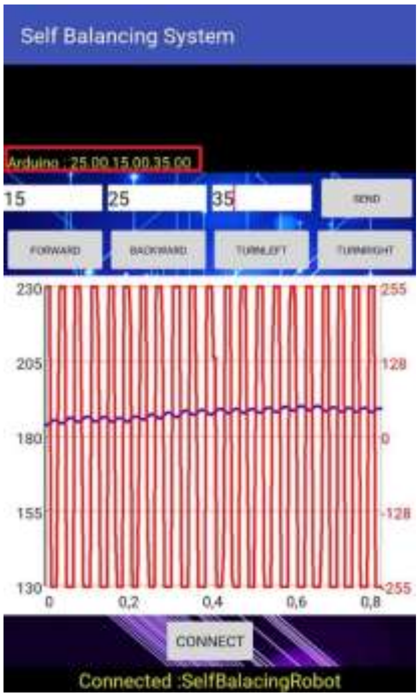


Figure 83 - Show status

VIII. References:

- Jinghua Zhong, Mechanical Engineering, Purdue University (Spring 2006). "PID Controller Tuning: A Short Tutorial" (PDF). Retrieved 2013-12-04.
- What's All This P-I-D Stuff, Anyhow? Article in Electronic Design.
- Ziegler, J.G & Nichols, N. B. (1942). "Optimum settings for automatic controllers" (PDF). Transactions of the ASME. 64: 759–768.
- Arduino.vn - Cộng đồng Arduino Việt Nam.
- Banlinkien.vn - Linh Kiện Điện Tử Minh Hà.
- 2011 IEEE 18Th International Conference on, vol.Part 1, no., pp.281-285.
- Accelerometer & Gyro Tutorial by Gadget Gangster.
- Liu Kun, Bai Ming, Ni Yuhua , Sept. 2011, Two-wheel self-balanced car based on Kalman filtering and PID algorithm, Industrial Engineering and Engineering Management (IE&EM).