Instrumentation, actuation and software development of Bipedal Robot

Lars-Erik Pangstuen[1] Edvard Merkesvik[1] Stian Johan Olsen[1] Kristoffer Meggelæ Pedersen[1]

[1]Norwegian University of Science and Technology, Trondheim, Norway

Introduction

An increasingly large part of modern society includes some form of robotics. In most cases this is fixed robots, however as new technologies are developed it is desirable to develop new robotics with higher mobility that can mimic human motion and operate outside the highly controlled environments where they are used today.
As such multiple groups at NTNU have over the last couple of years worked on a bipedal robot. This year has been no different. By working on the continued development of the robot, both the hardware and the software has either been fixed or upgraded.

The robot in question is an under-actuated biped that is restricted to move in a 2 dimensional plane. The robot consists of 2 stiff legs mounted to a torso at the "hip". With the mathematical models designed to work in two dimensions it is important to make sure to prevent lateral tilt, consequently each leg has two points of contact as if you would stretch the robot sideways when you add the third axis.

Hardware design

Early in the project it was discovered that several of the components had broken. As such it became a priority to make a new Printed Circuit Board(PCB). The board was made in order to accommodate all the functionality necessary for the robot, while making sure that nothing more would be damaged. This was also something previous groups had tried to accomplish. However because the problem still persisted, a new board was made. This board was made early and as such were made with a few assumptions. It includes PWM logic lever-shifting, OP-amp circuits, transistor circuits and encoder signal processing. Put simply the board was made with the extra functionality believed necessary to compensate for the things the Beaglebone Black lacks on its own.

Embedded system

The act of making a bipedal robot walk on its own is deceptively complex. It requires processing data from various sensors and using these in stabilizing algorithms. These algorithms do a large amount of mathematical calculations in order to balance the biped. As such it is necessary to control the robot using embedded hardware rather than using an external computer. There are several different components included in this system in order to make the biped walk as desired. It is also necessary for all the hardware to communicate effectively.
In order to make the robot walk as desired, these components have been worked on as parts of the embedded system:

- **IMU "Inertia Measurement Unit":** A device capable of measuring angle, angular velocity and its orientation.
- **Encoder:** A device that converts angular position of a rotating joint to a digital- or analog signal.
- **Motor:** The motors move the legs themselves. The embedded motor system consists of the servo controller, the gear and the motor.
- **Servos:** Four servos are used to move the actuators at the ends of the legs. This is to make sure the legs don't scrape the ground.

The system itself is controlled by a Beaglebone Black, which is a low cost open source single board computer. Both input and output signals are handled by this board, as well as any necessary interim computations.

ROS, Robotic Operating System

Not to be fooled by the name Robot Operating System (ROS) is not an operating system, but rather a framework for development of robotics. ROS distinguish itself from most other robot software by being open source and encouraging collaboration; this makes it ideal for education and research which has resulted in a quickly growing user base. As such it turned out to be ideeal for the purposes of making a bipedal robot.
There are several advantages of using ROS for this project, the main reasons were:

- **Modularity:** The modular nature of ROS is ideal for projects like this where components might be changed or added later on as the different nodes are independent of each other. This also means that you can use different programming languages for different nodes without problem.
- **Community:** The platform has a massive community meaning you can find packages that will take care of almost any task. The "ROS answers" forum is also a very helpful tool.
- **Visualization:** One of the main priorities for this project was to create a system for logging and real-time visualisation of the robot and ROS has great tools for this in the form of Rviz, rqt and the rosbag packages. Gazebo can also be utilized if simulation of the robot is needed.

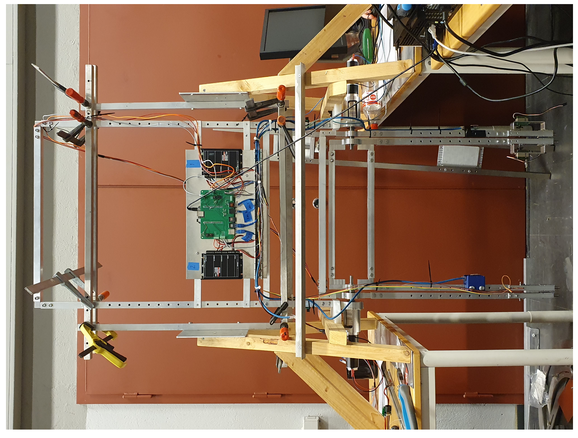

Fig. 1: Bipedal robot at the end of the project

Real time logging and visualization

After the dSpace system was abandoned in 2020 due to a ransomware infecting the computer that ran dSpace the robot lost all its visualization capabilities. This was a big hit for the project as a good GUI and real time visualization features makes testing and troubleshooting much easier. One of the main goals for the group towards the end was to make it easy to continue next year without spending too much time solving problems. As such a good GUI(Graphic User Interface) and a 3D visualization program were made.
Rqt is a built in framework in ROS that makes it easy to implement various GUI tools as plugins. One of the reasons why rqt is so powerful is its ability to interact with every node and topic in the system. This makes it easy to observe individual parts of the robot, we mainly used it to graph the encoder position. The interface created also allows the user to publish messages to the system. The following values can be changed using the interface:

- **IMU:** Values for sensor fusion for both PID- and Kalman values depending on which one is in use.
- **Motor:** The motor controls are implemented as sliders.
- **Servo:** Both sets of servos can easily be extended or retracted.

The visualization program used is called Rviz. Rviz allows for powerful 3d rendering in real time which can provide important insight into how the robot perceive itself and its surroundings.
The robot model used are based on three STL files made in fusion360 to describe the different parts of the robot. These are linked through 2 joints allowing dynamic movement. The robot publishes messages to update this 3D-model in real time. this is useful because it tells you where the robot thinks it is which can help us track down any errors in the measurements.

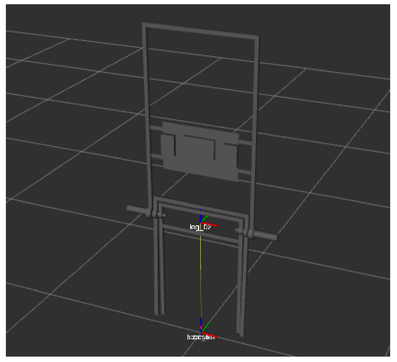

Fig. 2: Rviz piped model

Summary and conclusion

The main goal of this project was to continue the work on the robot to a meaningful degree. This meant finishing as many modules as possible. All the different parts of the embedded system were finished as planned. Even so there is still some functionality that needs to added later. In addition it became very important to make sure that it would be as easy as possible for the next group to continue developing the robot. For this goal better launch packages as well as a user interface and a visualization program were made. In addition a well-organized GitHub repository was made to be used for further development of the robot. As this is a project that builds on the work of several bachelor groups the importance of getting everything well documented can't be stressed enough. At the start of the project it was a goal to make the robot take a step or two. It did however become apparent that this goal was somewhat unrealistic. Mainly because the amount of problems with the robot were far more than expected. This led to changing the goal in order to focus on work that would bring more value in the long run. Even so it was agreed that the project ended up at a satisfactory level of completion. Choosing to focus on making the project easier to continue in the future seemed to be a smart decision. Hopefully the next group won't have to do as much troubleshooting and instead will be able to focus on the improvement of the hardware and software.