3 Analysis and Design

The development of the blue sleeve is a continuation of the blue MORF project, in a study about how the soft robotic breathing capabilities could influence the perception of the robot for humans in Robot Human Interactions. This study [**?**] was done by Mads Bering Christiansen, he designed the blue sleeve so it could simulate the MORF robot breathing.

My part of the blue MORF project is to add LEDs to the sleeve, I had to drill holes in the inner 3D printed shell of the sleeve. I chose to do this because the fabrication of a whole new sleeve would have taken too much time.
To analyse if the added LEDs will give any benefits to the design. I conducted a HRI experiment. To analyse the interactions the participants had with the robot, we will use the Robotic Social Attribute Scale (RoSAS)[1] and the 14 item subscale of the trust perception scale for HRI [2].

3.1 The initial blue MORF.

The blue MORF is a robot that consists of the standard MORF hexapod walking robot, with a added blue sleeve that can inflate and deflate its air pockets. As the standard MORF has a raspberry pi to control the walking, the blue MORF has a additional arduino (with motor shield) to control the 4 air pumps and 4 valves. The breathing rate is designed to be connected to the walking speed of the MORF. This communication between the pi and the arduino is done through a serial connection.

There are in total 4 ROS nodes working together:

1. **cpg_rbf dynamixel_node.py**
   This node is the driver to control the legs of the MORF. It listens to a rostopic and transform this data to the desired motor output. I changed nothing about this code, since the locomotion was not my focus. Without this node running, the legs will not move.

2. **rosserial_python serial_node.py**
   This node is necessary for the serial connection between the arduino and the raspberry pi. This sets up that communication channel. Since the beginning of this project, the arduino and the raspberry pi had problems with keeping their connections. The problem was to synchronize the data flow. It was not problematic because it would automatically recover

its connection really fast, however this error would still pop up in the terminal. I did not fully solve the problem, but I got the error to appear less frequent by changing the baud rate of this node, and the baud rate of the arduino from 115200 to 57000. This was the only thing I changed about this node.

3. **joy joy_node.py**
   This node is the driver for the remote controller. The controller we use for the MORF is a wireless logitech controller. It has a bluetooth dongle plugged in the raspberry pi. And the joy node handles the interpretation and sends it to a rostopic.

4. **cpg_rbf main.py**
   This node is the brain of the MORF. It listens to rostopic of the controller input and then calculates the right data for dynamixel node for the desired movements. It not only does this, it also calculates and controls the breathing of arduino. I changed a lot of code in this one, from the interpretation of the controller input to the additional data necessary for the arduino to control and synchronize the LED's with the breathing. I also made adjustments to the breathing code since there was a small bug.

In the beginning I was instructed to run all these nodes separately every time. This was okay to do a few times, but it got annoying after a while. I searched for a faster way of starting these nodes. And apparently there was a handy function for this called *roslaunch*. This needed a *.launch* file. That is a XML based file with the names, packages and parameters of the nodes you want to start. With this I could start the ROS environment, and all the nodes necessary with only one command. I made this file, this with a combination of a simple SSH connection to the MORF, I can start the MORF much faster.

Valve power problem

In the beginning the valves were not connected to the arduino. This was not a hard task to do, but for some reason I was not able to do so. After a while, I realised that the problem was that the valves did not get enough power to work consistently. This was strange because the researcher before me was able to do so. After some measuring I saw that the current provided by the shield was on the maximum of its capability. By piggy back soldering two L293D Motor Driver ICs on top of each other, the maximum power was increased and since then the valves worked consistently.

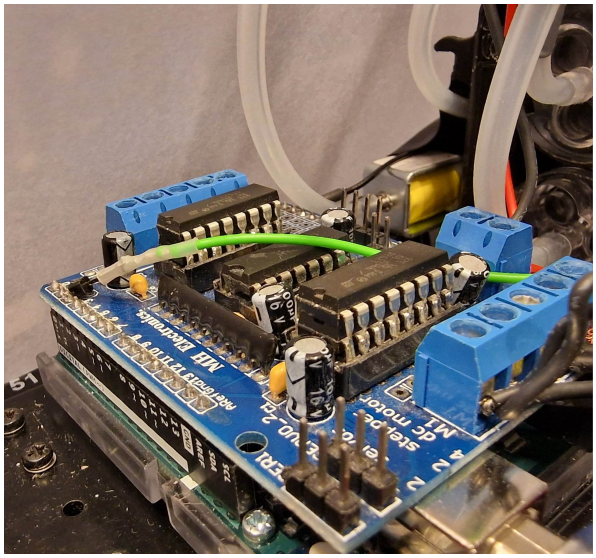

Figure 3.1: Piggy back soldered L293D 16-pin Motor Driver ICs on the motor shield.

3.2 Color configuration

Limitations to the color palette

The LEDs mounted are located underneath the silicone air pockets. This because the silicone would spread the light out through the whole air pockets and give a nice effect that does not hurt your eyes, if you look at it directly. However the sleeve is dark blue on itself, as you see in figure 3.2. This dark blue cover around the LEDs made it impossible to use the whole color palette that the RGB LEDs provided. It restricted the available colors to only green, blue (and dark red). Clear yellow, orange, pure red and white were made impossible. This was really unfortunate because, the human is used to pure red and orange signalling (e.g. traffic lights, left/right and breaking indicator lights on a car and motorcycles).

Another factor that influenced the looks of the lights drastically was that the thickness of the silicone was not perfectly even across the MORF. This may be the cause of the hand fabrication of the sleeve, or it may have been caused by over inflating the air pockets. However this meant that the same light setting on all the LEDs, would result in a different brightness intensity on every part of the sleeve. This made it even harder to make the lights look good.

1 Introduction

The blue MORF robot is a modular hexapod robot with a soft robotic sleeve mounted on its back. This project has passed through different developers, Master and PhD students. The last research done on the blue MORF is the addition of LED lights to the skin of the robot, and the study of how it effects the perception of robot by humans.

If you're new to the robot, the first thing you want to know is how to boot this hexapod. You have found the right documentation.

This will be a written documentation on how to boot it. There is also a video tutorial that guides you through the old process of booting the MORF. You may still use this method because it still works, or use the new method because it is faster. If you did not receive that video yet, you may ask your supervisor for it. But you will find the same (and more) information here. So you don't need to worry if you don't have it.

2 How to boot the MORF

2.1 Powering the MORF.

If you don't want the MORF to walk on your desk, **place it on its elevated stance.** Also be care full when powering the MORF that its legs have space to move. This is because it will subtract and raise it legs rapidly, after all the ROS nodes are activated, make sure It will not pull on any wires.

The head of the MORF is removable. Gently remove its head, and you will find a female XT60 connector.

> Connect a 22.2V power line to this male XT60 connector (see figure 1). I suggest a to use a power supply or a 6S LiPo battery.

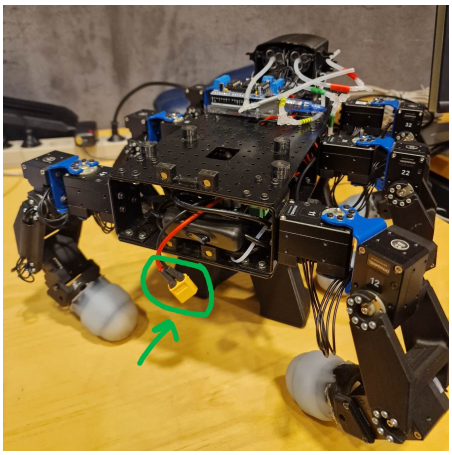

Figure 1: The male XT60 connector located at the head of the MORF.

Just connecting the power supply is not enough to power the MORF. There are 3 power switches you have to turn on. These three switches are located on the power board, near the head of the MORF. In figure 2 you can see two of the three switches. Number 1 is to power the arduino, number 2 is to power the raspberry pi. And the third switch is located on the other side of the same board and is to power the motors of the legs. You will feel that the switches mounted on the power board have 3 modes, but only the most utter positions have a function, so just flip them to the opposite side.

> Flip the three power switches on the board near the head to the opposite side. (The sequence of flipping these do not matter.)

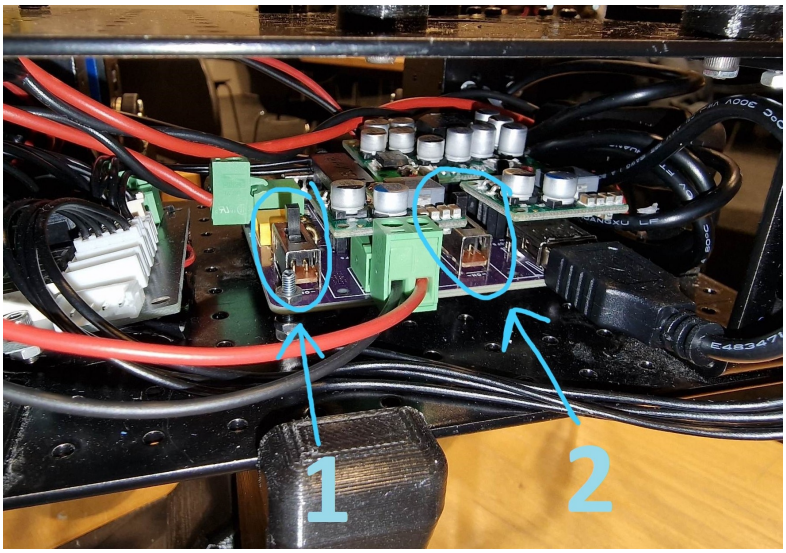

Figure 2: Two of the three switches needed to power the MORF.

Some lights inside the MORF will turn on, and you will see that the MORF is powered! Great! By powering the MORF, your raspberry pi started running its Ubuntu 20.4 Server. The following section will explain how to access this.

2.2 Accessing the Ubuntu Server

There are 2 main ways of accessing the Operating System. The first one is the most intuitive. But it requires you to have the right equipment. The second one is remote access from your computer.

2.2.1 Wired access

Find a monitor, keyboard and mouse and connect them to the raspberry pi. You can find the required ports on the raspberry pi, this is the silver box in the MORF. You can use the HDMI Micro port for the monitor and the USB ports for the keyboard and mouse. You will not have enough USB ports so u have to use a hub to do this. (The two USB ports on the raspberry are already occupied by the logitech bluetooth connector for the wireless controller

and the blinkstick square board for the led lights in the head.) If you have connected everything correctly, you should get the Ubuntu login screen on your monitor and you can navigate it with you mouse and keyboard.

Username: ubuntu
Password: 123456789

2.2.2 Wireless access

To set up a remote connection, you need a connection without a firewall. I used my phone as a hotspot because it's easy to find the IP address of the MORF. But an alternative is to use your pi to run a hotspot. (I did not have time to set this up myself, but this should be easier once everything is setup.) Once you know this IP address, and your device is on the same network as the MORF. You can use the following command:

A *SSH* connection:

```
ssh <username>@<IP_address>
```

Run this command in the terminal on your device with *'ubuntu'* as username and the IP address of the MORF. It will ask for the password, that is *'123456789'*. After this your terminal will have started a ssh connection. Every command you now enter, is a command you run in the terminal inside the MORF.

There is one risk with this method. If you are running the robot with this ssh connection, and your computer disconnects from the robot, the robot will stop reacting. If its pumping at the exact moment of the disconnection, then it will keep on pumping air, and it will stop reacting to your controller. This is really dangerous. You would have to disconnect the pumps or power of the MORF as fast as you can! Otherwise the air pockets will get so big they could pop. There may be a way of fixing this, but I don't have one yet. Just make sure that your computer does not fall asleep, or connects to another network.

2.3 Running the software

There are 2 ways to start the MORF software. The easiest and fastest one is the second. I suggest you use that one. It does everything mentioned in the first method, in one command and in one terminal. This makes the second one perfect for when you use the ssh connection. But behind the scenes of the second method is the first method, so to give you a good understanding I will also explain the first one.

2.3.1 Running the different ROS nodes individually

The ROS framework has 4 important components: nodes, topics, services and parameters. In this part, I'll tell you how you can run the different ROS nodes on the MORF.

Start a terminal. Execute: $ roscore

This is the same in every ROS application. The roscore command starts a ROS Master, a ROS Parameter Server and a rosout logging node which are the pre-requisites of a ROS system.

Every other node can only run in the system if the roscore is running and reachable. Read more on roscore on wiki.ros.org/roscore.

Now you have to run the different nodes required for the MORF. The *rosrun* command runs a node or set of nodes from an executable from any ROS package. The general syntax is:

```
$ rosrun package executable
```

These are the commands to run the MORF specific nodes. To start the next node after running the previous one in one terminal, a new terminal has to be open. You want these nodes to be running simultaneously.

```
$ rosrun cpg_rbf dynamixel_node.py
$ rosrun rosserial_python serial_node.py /dev/ttyACM0 _baud:=115200
$ rosrun joy joy_node
$ rosrun cpg_rbf main.py
```

2.3.2 Using roslaunch to do everything at once

Roslaunch is a tool for easily launching multiple ROS nodes locally and remotely via SSH, as well as setting parameters on the Parameter Server. Roslaunch takes in one or more XML configuration files (with the .launch extension) that specify the parameters to set and nodes to launch, as well as the machines that they should be run on. I wrote this file for you, so you can just use the following command from any directory on the blue MORF:

```
$ roslaunch cpg_rbf morf_blue.launch
```

This means that the *'morf_blue.launch'* file is located in the launch folder of the *'cpg_rbf'* package.

3 Programming on the blue morf

3.1 Easy file transfer

With this (and a ssh connection) it is possible to write code on you computer, and after less then 10 second run that code on the MORF.

***SCP* file transfer**

```
scp ./file.txt remote_username@remote_ip:/remote/directory
```

From a terminal on your computer, you can transfer a file from your computer to a directory of the MORF. You just have to locate the file and the directory. The username is as usual *'ubuntu'* and the command will ask for the password *'123456789'*.

3.2 Code storage

It is a good habit to regularly safe your code to a GitHub repository.

4 Controlling the MORF robot

We control it with a Logitech wireless gaming controller and these are the controls.

Start ⟶ restart
Y ⟶ increase speed
A ⟶ decrease speed
B ⟶ stop walking
X ⟶ Change color configuration
up ⟶ walk forward
left ⟶ soft turn left
right ⟶ soft turn right
down ⟶ walk backward
RB ⟶ start sequence
LB ⟶ flow air in

Note 1: Close to the middle of the controller, there is a *mode* button. The controller won't work if the its in the wrong mode. The right mode is the one where the green light is turned on.

Note 2: If you are making a turn to the *left*, and you immediately do a turn to the *right*. It will make an abrupt transition. You can get a smoother transition if you first go *up* and then *right*.

Note 3: If you see that the air pockets in the soft sleeve are getting too big. Quickly press the *reset* button (and decouple the valves if necessary). Or there will be permanent damage to the sleeve.

Figure 3: The wireless Logitech gaming controller.