Mälardalen University
School of Innovation, Design and Engineering
Västerås, Sweden

Project Course in Robotics and Embedded Systems,
DVA473 and DVA474

THE UNICORN PROJECT MANUAL

The Student Project Team

January 22, 2020

Table of Contents

1 2-D LIDAR Sensor OMD10M-R2000-B23-V1V1D **2**
1.1 How to connect it . . . 2
1.2 Setup LabVIEW for HTTP Commands . . . 2
1.3 Use ROS driver package for the R2000 . . . 3

2 Omd8000-r2100-r2-2 v 15 **4**
2.1 Serial Port Configuration . . . 4

3 DWM1001 **5**
3.1 Tag and Anchors Setup guide . . . 6
3.1.1 Configure a RTLS network . . . 6
3.1.2 Connect to DWM1001 via serial USB in Windows . . . 7
3.1.3 Connect to DWM1001 via serial USB in Linux . . . 7

4 Lift System **8**
4.1 Motor Controller . . . 8
4.2 Inductive Proximity Sensors . . . 8
4.3 LabVIEW VI's . . . 9

5 External and internal mounting **10**
5.1 Cable chain . . . 10
5.2 Valcro tape . . . 11
5.3 LED strip . . . 11

6 RoboRIO System Overview **11**
6.1 Startup . . . 11
6.2 ROS at startup . . . 11
6.3 Lift Statemachine . . . 11
6.3.1 0. Idle state . . . 11
6.3.2 1-4. Motor states . . . 12
6.3.3 5. Finished state . . . 12
6.4 R2100 Lidar . . . 12
6.5 UWB . . . 12
6.6 ROS Subscriber . . . 12
6.7 ROS Publisher . . . 13

7 How to communicate between roboRIO and ROS **13**

8 ROS simulation **14**

9 Boot-up **15**
9.1 Hardware Start-up . . . 15
9.2 Software Start-up . . . 15

10 Configuring and running RVIZ **15**

References **16**

1 2-D LIDAR Sensor OMD10M-R2000-B23-V1V1D

1.1 How to connect it

Go to this link: https://www.pepperl-fuchs.com/sweden/se/classid_53.htm?view=productdetails&prodid=86555#software. Download and install the required software.

The required electrical connections are two M12 connectors, one Ethernet and one power. In one end of the power cable there was no contact attached. The brown cable is 24V and the blue is ground.

Set the LIDAR to use DHCP and connect the Ethernet cable to the network for it to acquire an IP-address automatically.

You can log in to it and give it commands via the web or via Pepperl+fuchs' software.

1.2 Setup LabVIEW for HTTP Commands

To be able to run the HTTP commands in LabVIEW without errors some setup has to be done. First open NI MAX and access your roboRIO, right click on software and click "Add/Remove Software" as in fig 1. Log in to the roboRIO, default user name is admin. As default leave password empty.

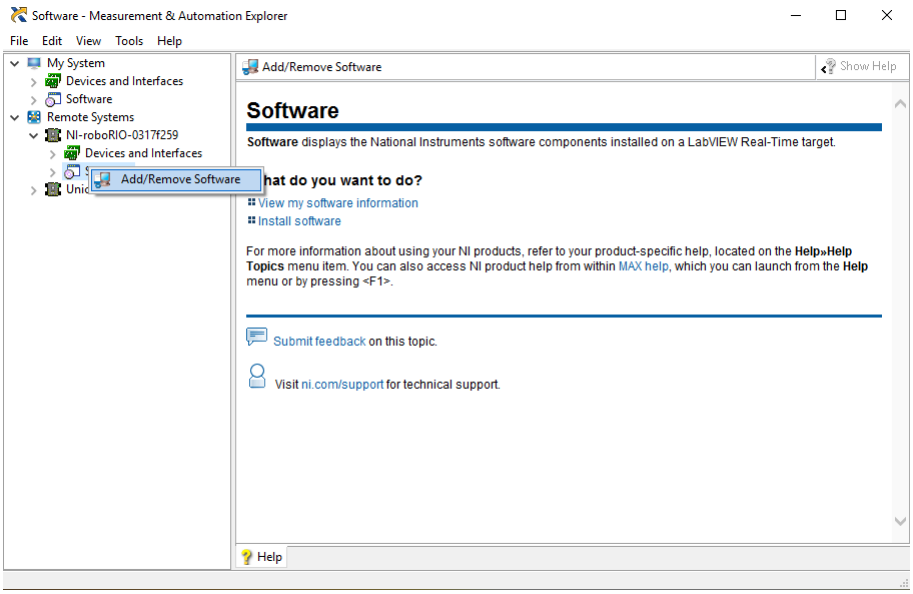

Figure 1: NI MAX window

Next, select NI roboRIO 16.0 under LabVIEW Real-Time 16.0.0 then click Next.

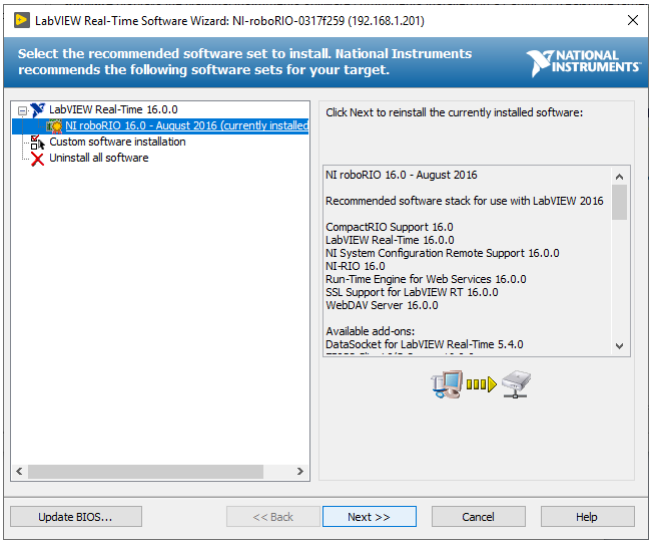

Figure 2: Real-Time Software Wizard

Next, find "HTTP Client with SSL Support 16.0.0" under Web Features. Mark this box down and click Next two times.

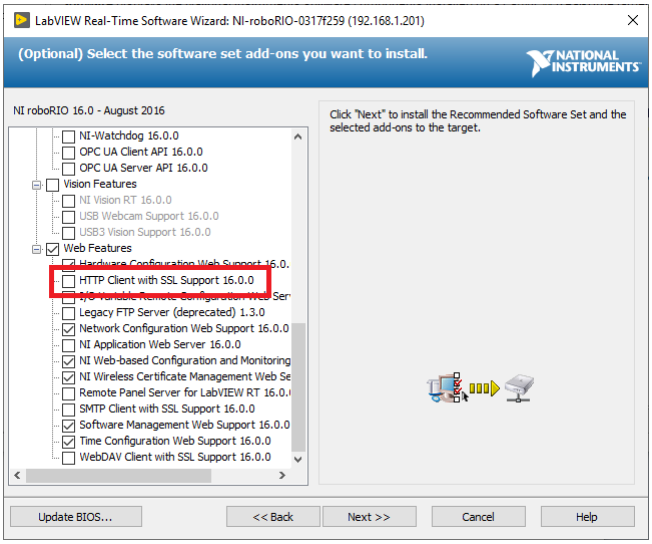

Figure 3: Install the marked software

The HTTP package software is now being installed and should work after the roboRIO reboots.

1.3 Use ROS driver package for the R2000

1. Clone or download the driver package from https://github.com/dillenberger/pepperl_fuchs into **/home/nvidia/catkin_ws/src**.
2. Compile it using **catkin_make**.
3. Run it with **roslaunch pepperl_fuchs_r2000 gui_example.launch**.
4. This will start RViz together with the driver and will display the measured output.

The driver package has been modified to fit the requirements for the Unicorn project. It can be noted that all changes made can be found by searching for the comment "Added by". The angle

range of measure is set to avoid recording the lift. The angle range can be calculated using the start angle and the number of samples per scan. For counter clockwise:

$$max\_num\_points\_scan = (3600000/samples\_per\_scan\_) * 3600000 - (3600000 + (start\_angle * 2))$$

The parameter **max_num_points_scan** works for protocol version 1.01 (R2000 firmware v1.20 or newer).

- A new launch file has been made named, **r2000Lidar.launch**. The differences between the two launch files are that the new one will not open RViz nor use the dummy_slam_broadcaster, then the node name and the topic name are changed.
- The watchdog is set to 'off' in the file, **http_command_interface.cpp**.
- The start angle is set in the file, **r2000_node.cpp**. It can be changed as one may please in the file, **r2000Lidar.launch**

2 Omd8000-r2100-r2-2 v 15

2.1 Serial Port Configuration

- Type: RS-232
- Baud Rate: 115200 kBd
- Bits: 8
- Parity: None
- Stop Bits: 1

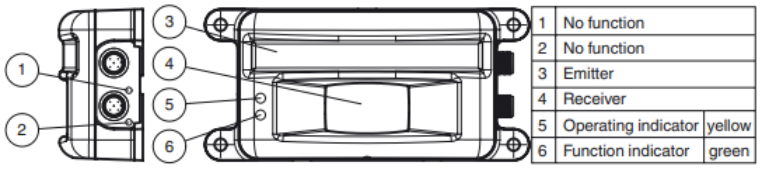

Figure 4: R2100 components from the datasheet.

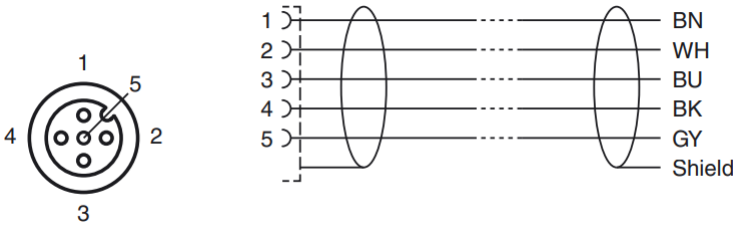

Figure 5: R2100 connection.

The RS232 connection is established using the RS232 connector on the roboRIO and the RS232 (grey) connector on the R2100. The Blue cable of the three in the RS232 is ground.

According to the datasheet the sensor has a measurement range of 0.2-8m. However, after testing, it could be established that the sensor could provide somewhat reliable measurements as close as 0.1m.

3 DWM1001

UART is used to establish a connection between the DWM1001 and the RoboRIO.

UART communication is asynchronous and has therefore no need for a clock signal for synchronization between data bits transmitted and received. Instead it uses start and stop bits added to the data packages which are used by the receiving UART to determine were to start and stop reading. The data bits are transmitted and read at a specific frequency called, Baudrate. A parity bit can be added to the package sent which is used for error handling. Depending on if the parity bit is even, 0, or odd, 1, the data 1-bits in the data frame should sum up to even or odd, respectively, for there to be no error.[**?**]

The configuration for the communication between the DWM1001 and RoboRIO is;

$$115200/8N1$$

This means that it uses a Baudrate of 115200 bps, 8 data bits in the data frame, no parity bit and 1 stop bit.

The reasons why UART were chosen over SPI are that it fits our requirements, it demands only two wires and no clock signal is needed.

Following tabular shows how the physical connection is set up.

DWM1001 Pins			roboRIO Pins		IO Iso Pins	
Connector J10	Function	Wire color	MXP	Function	Connector J10	Function
Pin 6	GND	Brown/white	Pin 12	GND	Pin 21	GND
Pin 8	RXD	Blue/white	Pin 14	TXD	Pin 22	TXD
Pin 10	TXD	Orange/white	Pin 10	RXD	Pin 23	RXD

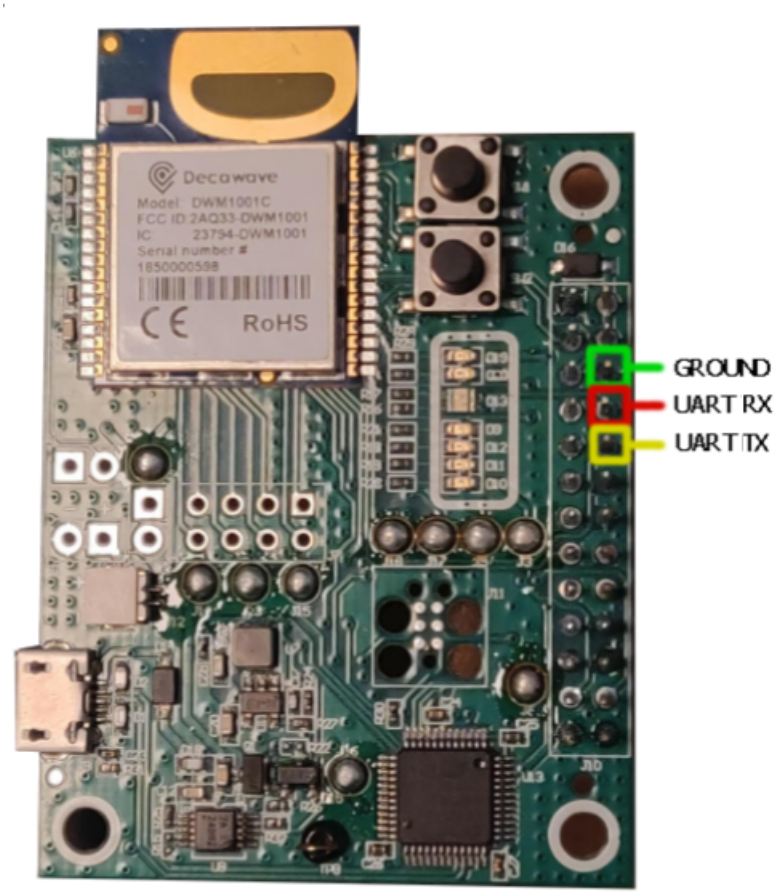

Figure 6: How to connect with UART to DWM1001

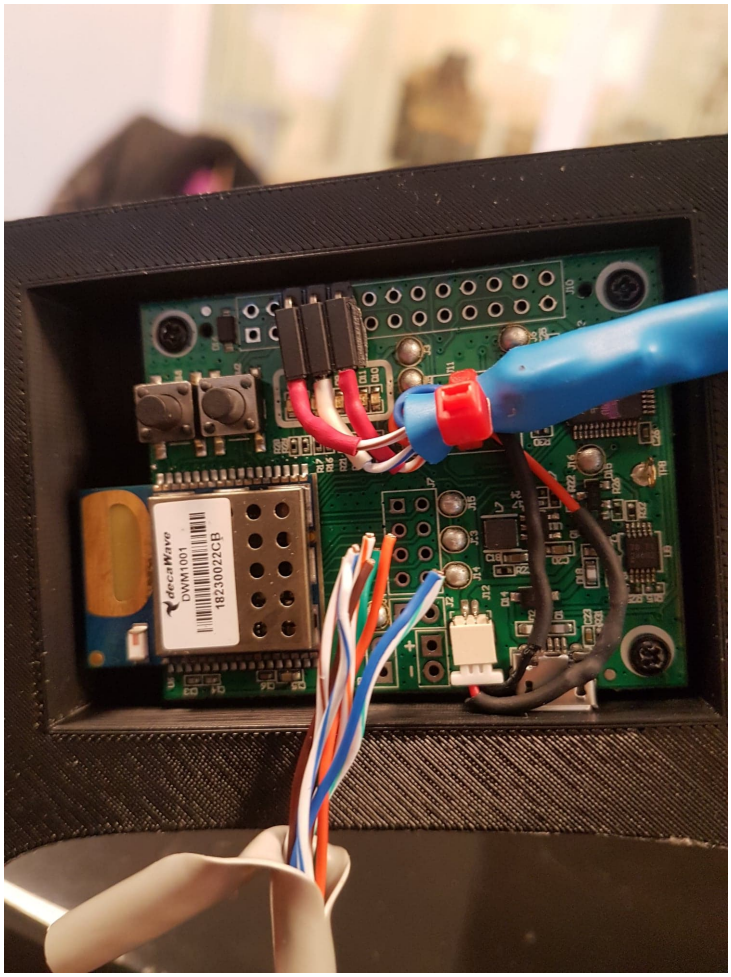

Figure 7: How the physical connections are wired.

- The orange wire is DC power.
- The blue wire is ground for power.
- The brown/white wire is UART ground.
- The blue/white wire is UART RX.
- The orange/white wire is UART TX.

3.1 Tag and Anchors Setup guide

3.1.1 Configure a RTLS network

[?]

1. Download the latest version of the DRTLS Manger from the Google Play store (Android) or from https://www.decawave.com/
2. Start up a DWM1001 and start the Android App
3. The DWM1001 has an ID number displayed in the App. Every anchor/tag is marked with it's induvidual ID number
4. Start as many devices desired for your network and make sure they are identified by the App before starting to configure the network

5. Create a network and name it
6. Assign your devices to your network Now you can configure your network. At this stage all devices have the same configuration
7. Set the first devices as an active anchor. The network requires at least one anchor to be an initiator
8. Set one device as an active tag
9. Set the rest of the devices as active anchors
10. Set up your anchors in your testing area
11. Go to Auto positioning in the up-right corner of the App
12. The anchors shall be arranged anticlockwise. If you don't find the anchors go back and edit the anchors by giving them some random initial position (not the same)
13. Press **MEASURE**
14. The anchors positions can be previewed by pressing **PREVIEW**
15. Test your network under **Grid**

3.1.2 Connect to DWM1001 via serial USB in Windows

1. Download/Open Terra term or Putty
2. Connect to the USB COM port
3. Goto Setup - Serial port...
4. Setup

```
Speed: 115200
Data; 8 bit
Parity: none
Stop bits: 1bit
Flow control: none
```

5. You should now see the DWM1001 user interface, type '?' for help

3.1.3 Connect to DWM1001 via serial USB in Linux

1. Download and install Minicom:

```
sudo apt-get install minicom
```

2. Open minicom configuration menu:

```
sudo minicom -o -s
```

3. Go to *Serial port setup*
4. Go to Serial Device by pressing 'A'
5. Change the serial device to your USB port connected to the DWM1001 and press enter. E.g: /dev/ttyUSB0

6. Go to Hardware Flow Control by pressing 'F'. It shall be configured as 'No'
7. The rest of the setting shall be 8N1.
8. Press 'esc' to exit
9. Go to 'Exit' to start the connection with the specified configuration
10. You should now see the DWM1001 user interface. Type '?' for help

4 Lift System

4.1 Motor Controller

The 12V geared brushed DC motors with a torque of 0.2 Nm:
https://www.elfa.se/Web/Downloads/55/07/05445507.pdf

Last year's iteration used an isolated switching step down/up converter to convert the voltage from the PDB from 15V to 12V. This will not be needed because of the new PDB.

The motor controller is an L298N dual bridge DC stepper controller board: https://www.velleman.eu/products/view It's connected to the roboRIO according to the tabular below. The two DC motors are connected to ports MOTORA and MOTORB. The motor controller can serve motors with input voltage between 5 VDC to 35 VDC via the VMS port. To drive a DC motor with this controller the enable port for the motor must be enabled by setting it to HIGH. Then the control pins need to be set. To drive it counter-clockwise, the IN1 and IN3 ports are set to HIGH. To drive it clockwise, the IN2 and IN4 are set to HIGH.

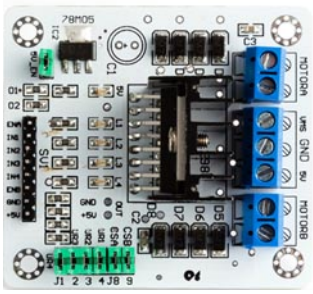

Figure 8: L298N dual bridge DC stepper controller board

4.2 Inductive Proximity Sensors

The linear position sensors mentioned above will be replaced by four inductive proximity sensors from Pepperl+fuchs. The data sheet for the sensor can be found under this link: https://files.pepperl-fuchs.com/webcat/navi/productInfo/edb/272511_eng.pdf?v=20190805165410. The sensor should be connected to analogue in and 5V on the roboRIO. The BU- cable should, besides being connected to analogue in, be connected to ground via a 3.6 kohms resistor or similar. This gives an output of approximately 2.5V.

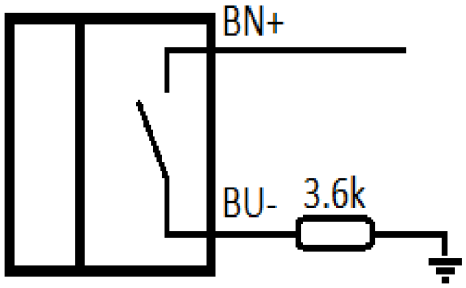

Figure 9: Inductive proximity sensor configuration.

roboRIO MXP		
Sensor	Pin MXP	Voltage
Motor Controller (control pin 1)	11 (DIO0)	3.3 - 5.0
Motor Controller (control pin 2)	13 (DIO1)	3.3 - 5.0
Motor Controller (control pin 3)	15 (DIO2)	3.3 - 5.0
Motor Controller (control pin 4)	17 (DIO3)	3.3 - 5.0
Motor Controller (Enable A)	19 (DIO4)	3.3 - 5.0
Motor Controller (Enable B)	21 (DIO5)	3.3 - 5.0
Inductive proximity 1	23 (DIO6)	3.3 - 5.0
Inductive proximity 2	25 (DIO7)	3.3 - 5.0
Inductive proximity 3	27 (DIO8)	3.3 - 5.0
Inductive proximity 4	29 (DIO9)	3.3 - 5.0
UWB TX	10 (UART.RX)	3.3 - 5.0
UWB RX	14 (UART.TX)	3.3 - 5.0
UWB GND	12 (DGND)	3.3 - 5.0
UWB Supply	4 (AO1)	2.8 - 3.6
UWB Supply GND	6 (AGND)	2.8 - 3.6

4.3 LabVIEW VI's

- LiftStateMachine.vi
  The final statemachine for the lift. This VI work as the flowchart shown above in figure **??**. Without ROS the statemachine is idle until the user pushes the USER button on the RoboRIO.

- ManualTestVI.vi
  Enables the user to manually run the motors via the motor controller, it also measures the input to AI0 on the RoboRIO.

- SwitchSensorReader.vi
  Reads every switch on the lift system and outputs their values.

5 External and internal mounting

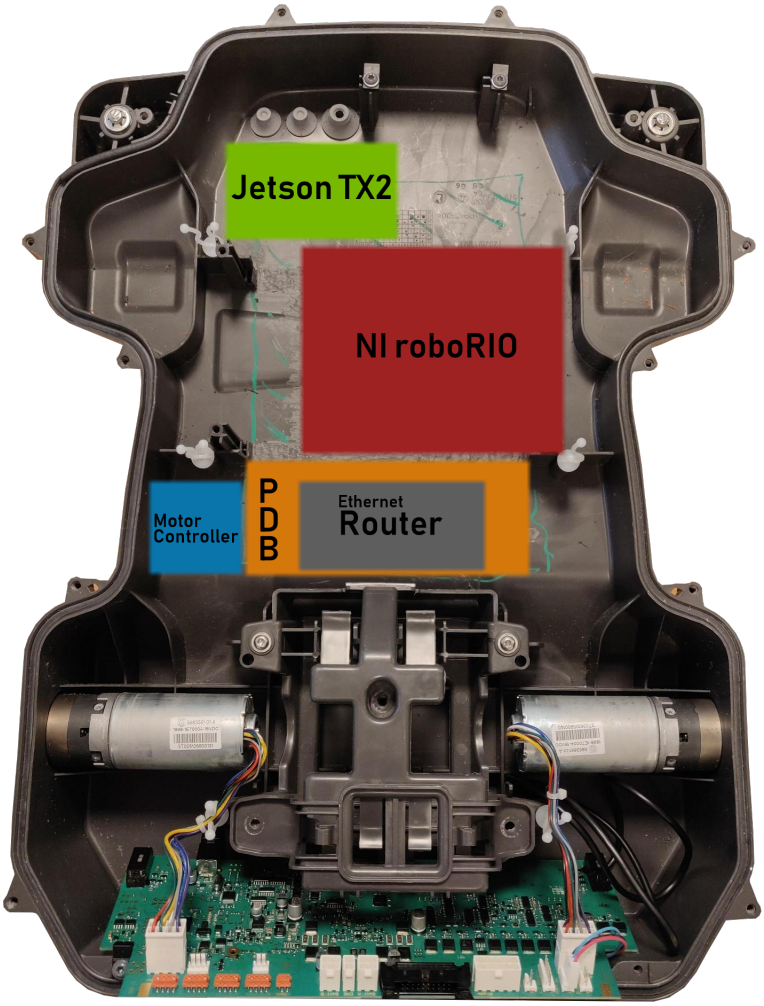

Figure 10: Hardware mapping of the robot internals.

5.1 Cable chain

The lift system has moving parts and several cables connected to sensors on these parts. This can be problematic due to the risk of cables getting stuck and becomes damaged. To minimize this risk the cable chain in Figure 11 can be used. The cables will be carried inside the cable chain between the moving and the static parts. Due to the lack of time it has not been bought yet. The chain are 10x15mm and 0.5m long.

Figure 11: The cable chain for the moving part of the lift.

5.2 Valcro tape

Valcro tape is used to connect the mounts of the different devices to the inner floor of the robot.

5.3 LED strip

The LED strip is mounted on galvanised banding, using cable ties, which is mounted in the chassis using screws and silicon. A hole were made so the cables could be wired into the robot to the roboRIO.

6 RoboRIO System Overview

6.1 Startup

On startup a custom FPGA bitfile is loaded first to make sure the FPGA only uses the DIO ports of the roboRIO. Then all global and local variables are reset to their default values. The ROS master ip is set via a constant string and the necessary ROS variables are set. All systems are then started in parallel while loops until the emergencyStop global variable is true.

6.2 ROS at startup

The publisher and subscribers can sometimes initialise with errors which makes the whole system malfunction as ROS is a vital part. To solve this a check is made inside the publisher/subscriber loops if there is an error. If an error occurs a timer starts and the publisher or subscribers are re-initialised. Simultaneously a check is made in parallel if the master can be reached. If not the timer starts. When the timeout count is reached the system times out and is restarted.

6.3 Lift Statemachine

The lift statemachine consists of 6 states, state 0 is the idle state, state 1-4 is the different motor states, state 5 is the finished state. The states are controlled through a local variable called $liftState$. The current state is also published on ROS.

6.3.1 0. Idle state

In the idle state the lift is idle waiting for the global variable $runLiftStatemachine$ to be true. During this state the motors are turned off and the COMM LED on the roboRIO is lit GREEN.

6.3.2 1-4. Motor states

The different motor states is the process of picking up a refuse bin. State 1, the motor A drives forward towards the bin. State 2 the motor B drives up to catch the bin. State 3, the motor A drives backwards to move the bin out. State 4, the motor B drives the bin down to finalise the lift.

During each state the COMM LED on the roboRIO blinks ORANGE/RED.

6.3.3 5. Finished state

In the final state the current lift state is published as a five which tells the TX2 state-machine that the lift is complete. A wait of 1000ms makes sure the lift is not restarted before the TX2 state-machine have had time to switch state.

6.4 R2100 Lidar

The R2100 Lidar starts in an idle state waiting for a global variable to be turned to true by the ROS subscriber. Running the code in LabVIEW enables the option to start the Lidar without a ROS message.

When the Lidar is in Idle mode the MODE LED is lit GREEN.

When the TX2 state-machine enters the reversing state a command to start the back lidar is given to the R2100 controller and the MODE LED is lit Orange.

The roboRIO then sends a bit array command over UART to the lidar telling it to do a scan and send the data. A timer is engaged to give the Lidar time to sent the 50 byte data response. The data is then received on a UART port on the roboRIO.

A check is made to see if there is any message received. If there is no data received a timeout timer starts. If there is no data received within 5000ms the RADIO LED is lit RED. A check is also made on the received data. Byte 2 and 3 have a constant value, if any of these values deviates from what is predicted the message is discarded and the timeout timer starts.

If there is data received the bits are formatted so 11 float distance measurements remain to be read. The message published to ROS is a average of the 5 sensors located in the middle.

Sometimes when first starting the R2100 Lidar the incomming bit array can be out of sync. This is automatically corrected when the program see the desync it adjusts the wait times and eventually the sync is correct.

6.5 UWB

The UWB system starts by sending a RESET command over UART to the DWM1001 to make sure the no previous commands are running. It then enters two parallel loops, one to send two carriage return commands that the DWM responds will responde to, and the other loop to check for the response. The DWM will respond when it is up and running after the reset.

When the DWM responds and is up and running these loops a exited and two new loops are entered. The first loop sends a position request command to the DWM and the second loop listens for the response. The response is formatted in the following way, "x:111y:222z:333qf:44". Each position value is extracted and built into a cluster with x,y,z float values.

6.6 ROS Subscriber

The ROS subscriber listens to several topics. These are:

- */TX2_unicorn_state*
  Listens to what state the unicorn is currently in. Tells the back lidar to start when it is in a reversing state. Tells the lift to start when in lifting state and tells the led strip which state to light up.

- */TX2_localplanner_intention*
  Listens to what direction the local planner currently wants to move to send to the LED strip when navigating.

Each subscriber produce a message that is put into global variables to be read in the sub-VIs.

The subscriber checks every loop if there is an error on the ROS read nodes. If an error occurs the subscriber loop is exited and the nodes are reset.

6.7 ROS Publisher

The ROS Publisher publishes one topic, this is:

- $/RIO\_publisher\_masterMessage$

The ROS publisher publishes a custom message called $/msgsBuilder\_MasterMessage$. This message consists of a built JSON string from several status variables indicating the current status of all the systems/sensors on the Unicorn. These variables are:

- $liftStatus$
- $lidarBackStatus$
- $uwbStatus$
- $lightStatus$

It also consists of the current lift state, UWB position and average distance on the back-lidar. This message is published on the topic $/RIO\_publisher\_masterMessage$.

The publisher checks every loop if there is an error on the ROS write node. If an error occurs the publisher loop is exited and the node is reset.

7 How to communicate between roboRIO and ROS

In order to establish a communication between the roboRIO and ROS some software must be installed. In this project a github software were used, ROS for LabVIE, which can be found under this link: https://github.com/tuftsBaxter/ROS-for-LabVIEW-Software. Download the repository as a zip-file. Un-zip it and go to the sub-folder, **ROS-forLabVIEW-Software-master**. Copy the folder, **ROS for LabVIEW Software**, into user.lib in your LabVIEW directory.

For the roboRIO to work with a master that isn't running on the unit itself, a source distribution needs to be created. Follow the guide on how to do it in the github repository found under this link: **ROS-for-LabVIEW-Software/ROS for LabVIEW Software/Help/myRIO and roboRIO Help.pdf**.

Make a real-time application by right-clicking the Build Specifications folder on the roboRIO. Don't forget to right-click the Build Specification folder and Build all before deploying your real-time application together with the source distribution onto the roboRIO.

It should look something like in Figure 12 when it's done.

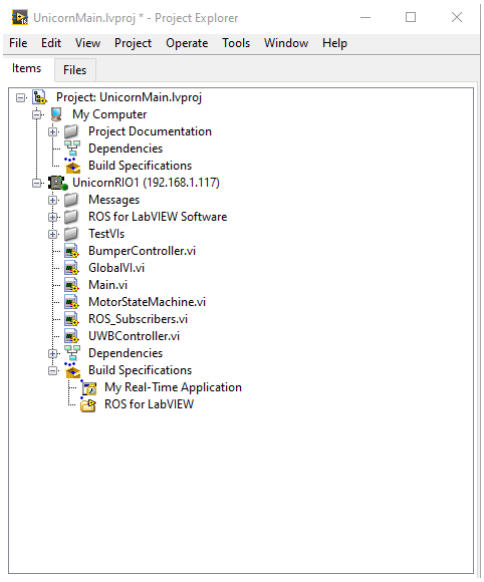

Figure 12: How your project in LabVIEW could look like when everything is done.

8 ROS simulation

Please read the full simulation guide before starting the simulation.

To successfully run a simulation of the unicorn robot with gazebo do the following:

Step 1: Open a new terminal.

Step 2: Type: **roslaunch unicorn unicorn_sim.launch**

If the simulation fails to start, just try again.

In the launch file (unicorn_sim.launch), under the section "Launch gazebo world", different maps can be loaded.

Map without moving obstacle:

1. value="$(find husky_gazebo)/worlds/clearpath_playpen.world"

Map with moving obstacle have two options:

1. value="$(find unicorn_slam)/worlds/clearpath_playpen_moving_obstacle.world"
2. value="$(find unicorn_slam)/worlds/testworld_moving_obstacle.world"

The playpen world with moving obstacle also includes several static obstacles. The testworld with moving obstacle includes no static obstacles.

How to enable moving obstacle in Gazebo:

To properly launch a map with a moving obstacle a script needs to be activated and a path needs to be added to the terminal. Observe that the following commands needs to be written in the SAME terminal as the one eventually starting the simulation.

1. cd ~/catkin_ws/src/UNICORN-2019/gazebo_animatedbox/build && make
2. export GAZEBO_PLUGIN_PATH=‘pwd‘:$GAZEBO_PLUGIN_PATH

Any of the worlds with moving obstacle can now be successfully started from the same terminal.

9 Boot-up

The following subsections explains how to power-up and start the UNICORNs robot.

9.1 Hardware Start-up

To power up the robot, simply flip the power-button located below the front side of the robot.

9.2 Software Start-up

The primary means for interacting with the robot is by connecting to the Jetson TX2 via SSH using the following command:

```
ssh nvidia@10.0.0.2
```

with password `nvidia` when connected to either of the robot's WLANs(`UnicornLAN2` or `UnicornLAN5`). It is recommended that this process is scripted to ease the connection process.

To start the robot three programs/files must be launched/runned via ROS in a specific order, each in their own terminal respectively. Meaning you need to open a terminal, login to the TX2 via SSH and then execute one of the commands below, each in a seperate terminal. The following commands start said programs via ROS:

```
roslaunch unicorn main_2019.launch
rosrun unicorn unicorn_statemachine
rosrun unicorn_cmd_interface unicorn_cmd_interface
```

For the robot to actually function only the first two commands are required. However, in order to issue commands to it, the interface must be running. The available commands which can be issued via the Command Interface Node is displayed in the terminal from which it was started.

10 Configuring and running RVIZ

An optional method to issue move commands is via RVIZ [1] if it has been configured on the remote computer.

To configure and run RVIZ to visualise sensor data and issue move commands remotely the following commands must be executed on the remote machine.

```
export ROS_MASTER_URI=http://10.0.0.2:11311/
export ROS_IP=<remote machine IP>
rosrun rviz rviz
```

Further information about using RVIZ remotely to monitor and control a robot can find at this link: http://wiki.ros.org/robotican/Tutorials/Remote%20monitoring%20and%20control

[1] http://wiki.ros.org/rviz

References