Final Documentation for the humanoid foot demonstrator of Group 2

This document provides vital information and installation guides of all implemented components used in the humanoid foot. The functionality of the foot is simplified to one active pitch movement controlled with a dc motor a passive roll movement and a passive roll movement of the toes. The foot movement is detected with three IMUs, in addition an encoder is used to record rotation after the gearbox and force sensors are used to detect ground contact. The overall control is done witch a Simulink model run on a raspberry pi. The sensor data is processed with an Arduino Uno.

Table of content

Design

Tasks distribution

The following table gives a brief overview of the task distribution in our group, after the design process was done.

TASK	
ASSEMBLY & TESTING	Christian Brendes, Doruk Utkan, Emin Nakilcioglu, Kevin Deutmarg, Marvin Gravert, Stephan Hansen, Till Staude
CAD-DESIGN	Doruk Utkan, Emin Nakilcioglu, Till Staude
ENCODER	Till Staude
FORCE SENSOR	Stephan Hansen
IMU	Marvin Gravert
MOTOR CONTROL	Christian Brendes, Kevin Deutmarg
REFERENCE SIGNAL	Gunnar Maerker
SIMULINK MODEL	Christian Brendes, Kevin Deutmarg, Marvin Gravert, Stephan Hansen, Till Staude

Requirement list

The requirement list was the result of an iterative design process and the base for assumptions and calculations in the part design/selection process. For further information take a look in our midterm presentation “20180518_HumRob_Midterm”.

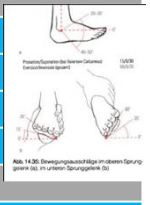
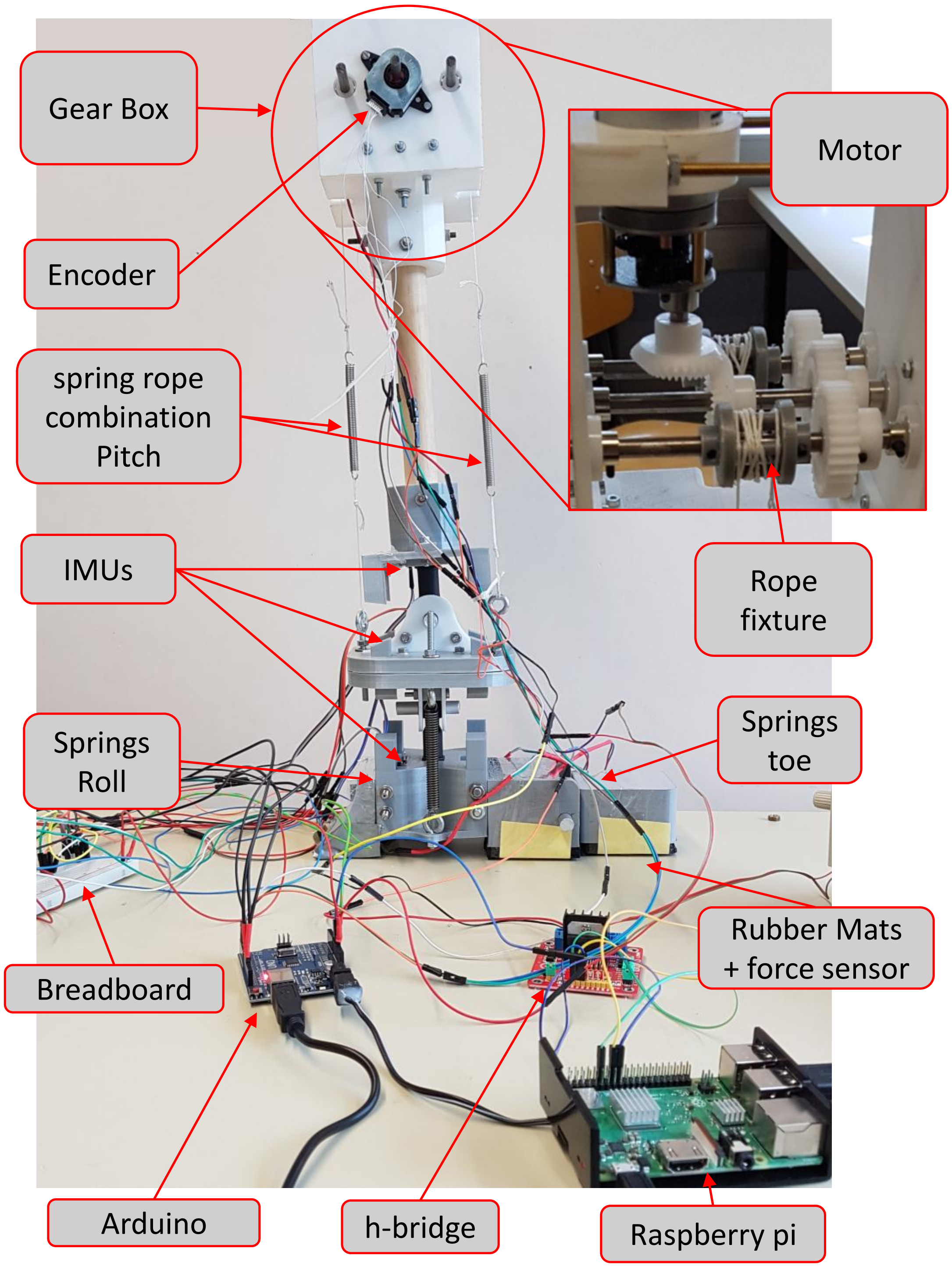
Nr.	Bezeichnung der Anforderung	Zahlenwert min	exakt	max	Einheit (phys.)	Art der Anforderung
1	**Geometric Requierment**					
1	Area of Foot (rectangular)			741	cm²	* Prototype Factor
2	Weight foot		0,4		kg	20kg human 2%
3	Toes or Simplification					
4	Leg adapter					
2	**Kinematic Requirement**					
1	Pitch (ankle)	-30		20	Deg	
2	Roll (ankle)	-30		60	Deg	
3	Bend Toes / Forefoot					
4	Actuator speed					
3	**Forces**					
1	Balance					
2	Shock absorption					optional
3	Maximum Torque (ankle)			30	Nm	
4	Total robot weight	10		20	kg	
4	**Energy**					
1	Energie preservation: pot ←→ kin ←→pot					

Illustration of the demonstrator

Humanoid foot demonstrator Group 2



Hardware part list

The following list shows the implemented parts for this demonstrator.

Partname	Function
Raspberry Pi 3 B+	Microcontroller for the motor control setup
H-Bridge, Iduino L298N	DC motor controller
Arduino Uno R3	Microcontroller for processing sensor data
Teensy 3.6	Microcontroller for processing IMU data
12V DC Motor	Actuator for the demonstrator
1x Encoder	Detect rotation of rope fixtures
3x IMUs (MPU9250)	Detect roll and pitch movement of foot
4x Interlink force sensors	Detect ground contact
2x Spring tilt	
2x Spring roll	
2x Spring toe	
1x Bevel gear	
3x Spur gear	
1x A/D converter MCP3008	
2x Amplifier LM 393P	
Rubber mat	Adding some damping to ground contact
IGUS bearings (no. BB-626-B180-30-ES, GSR-06, KBRM-10 OK, GBM-10, GSR-10, GFM-101115-03)	
12V Power Supply Unit	Power supply for the DC motor
1x Breadboard	Used for quick wiring
RJ45 Cable, USB AtoB cable	
Wires, Resistor …	Additional electrical components
Steel shaft, screws, …	Mechanical components for assembly

Assembly Instructions for the current state

The foot is stored half-assembled right now. All small wires are connected and most mechanical parts are in place. If you want to put the foot together, only the following steps are needed:

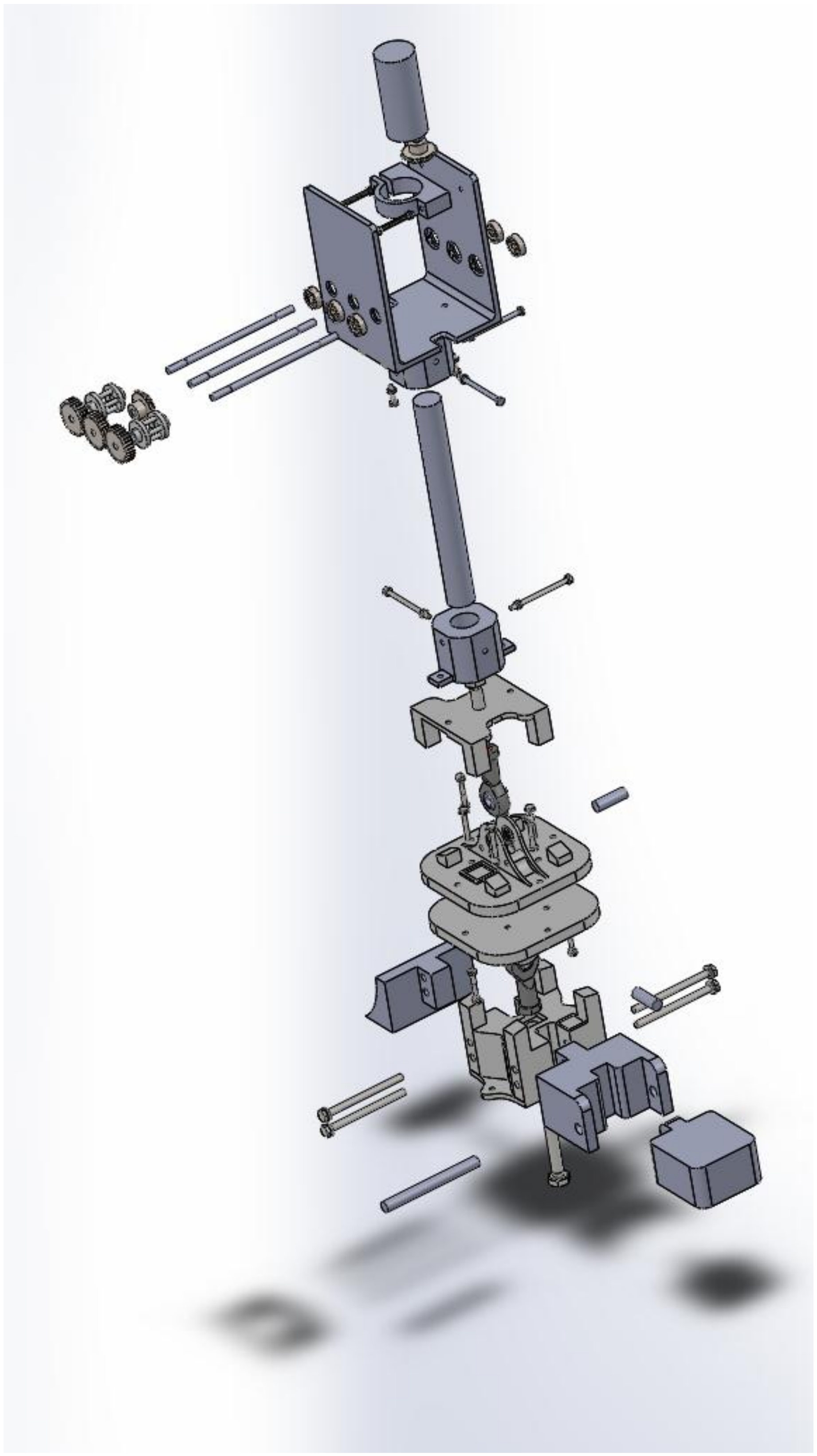
Mechanical Assembly

- put together the bearing and shaft for pitch
- attach roll springs, tight the screws such that roll is in balance
- attach pitch springs, tight the screws that pitch is in balance

Electronic Assembly

- connect the Arduino to the Raspberry via USB (B to A) using the upper left USB port
- connect the Teensy to the Raspberry via USB (Micro-B to A) using the lower left USB port
- connect the Arduino and Raspberry to their power supplies
- connect the H-bridge to a power supply (12V)
- connect the Raspberry via Ethernet to your PC running Simulink

Exploded view of the CAD Model



Model for Pitch Behavior

We created a simple model for the pitch behavior which may help as a start for control purposes.Please find the illustrations and calculations in a separate file: Attachment_Pitch_Model.docx

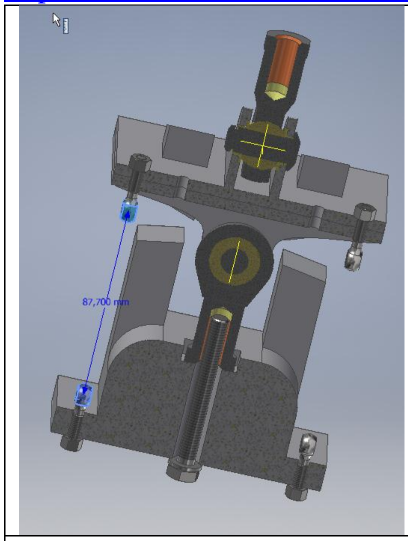
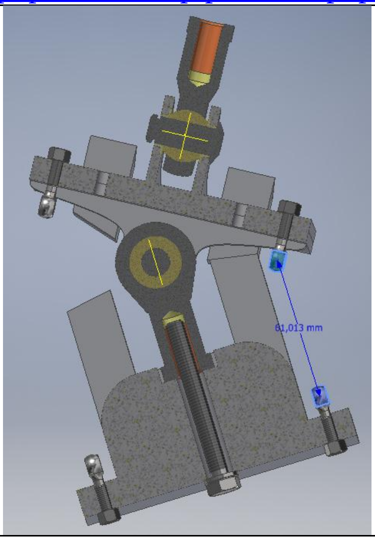
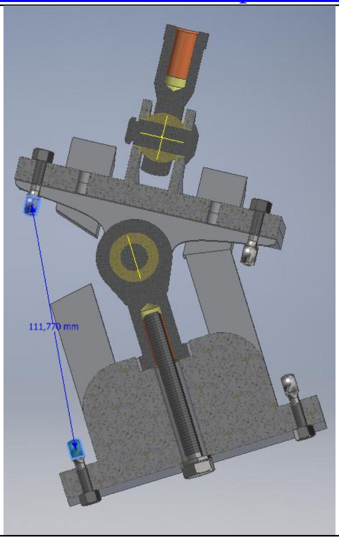
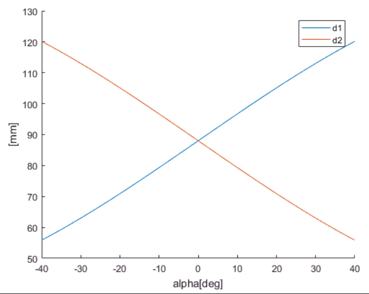
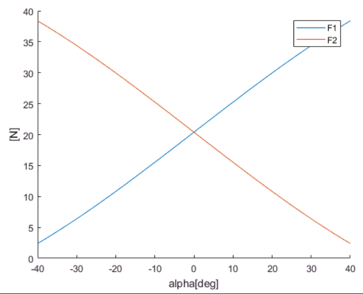
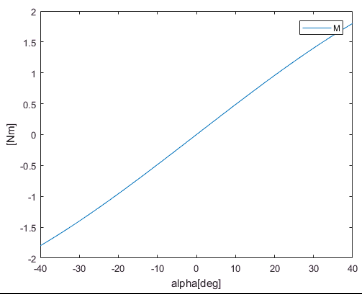
The model has not been tested in further detail but it has been implemented in matlab/Simulink and can be found in the folder Attachment_Pitch_Model

Selection of components

Selection of Roll Springs (Extension Springs)

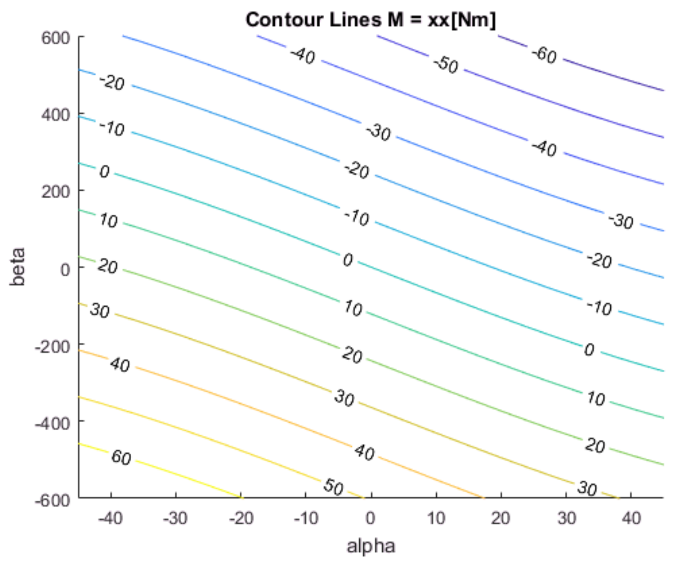
Aim: Find springs such that the roll of up to +/-30° happens with relatively little resistance (1-2Nm) estimated by body weight and leverage when ground contact is not centered below the foot. [I'd recommend to reestimate the value in a next iteration]

Link to shop:
https://www.febrotec.de/content.php?seite=shop/produkte.php&details=16226&hauptrubrik=23

		
Zero Position	At 30° (max angle)	At -30° (min angle)
		
Spring length over angle	Force in spring over angle	Moment of both springs about rotation axis over angle → see what angle is achieved by which external moment

Find in files: RollSpringDiagram.mlx → Code for calculations

Selection of Pitch Springs (Extension Springs)



For selected spring rotation of motor (beta) and pitch angle (alpha) and corresponding moment over pitch axis. (These values are independent of the roll angle!)

Note: We selected rather hugh springs as we initially planned to support big moments which may be necessary for walking (30/40 Nm). During the design we finally assumed our prototype would not manage the big pretension forces caused by these springs, so finally they were not used. For the next iteration in design, either design a more stable component (problem: expensive, harder to manufacture) or choose smaller springs and assume smaller load values (i.e. moments).

Find in files: PitchSpringDiagram.mlx → Code for calculations

Troubleshooting:

The selected (and bought) springs were way too massive to be implemented in the current system, since the prototype could probably not withstand the loads assumed for a full (child sized) human. So we did finally decide to not use the spring but users we got in the workshop. In the next iteration change the size of the components (some were oversized, others were too small) to fit the expected loads.

Selection of Actuator

To select a suitable actuator for the pitch movement initial calculation were necessary to get a number for the motor power and rotational speed. The following table contain the Power calculation of our motor. Therefore simplified assumptions of the overall dimensions of our foot design were made, illustrated in Figure 1.

Foot Moment	30 Nm -> requirement list	
Foot Mass	3000 g (see figure .. for dimension)	Density 1.5 g/cm³ 3D-Printer Material
Inertia	= 0.0275 kg*m²	
Spring Force	30 Nm / 0.05m = **600N**	

Gear Wheel Moment	600 N * 0.0075 m = **4.5 Nm**	
$\dot{\beta}$	100 [mm/s] / 7.5mm = **13.33 rad/s**	100 mm/s [source: doi:10.1098/rspb.2011.1194]
Motor Power P	4.5 Nm * 13.33 rad/s = **60W**	

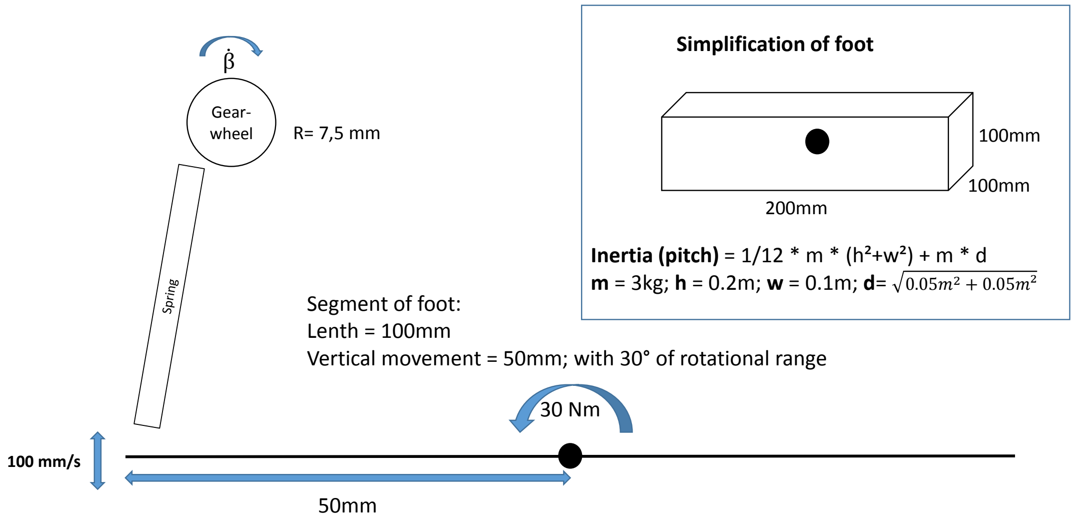

$\dot{\beta}$
Gear-wheel
R= 7,5 mm
Spring
Simplification of foot
100mm
100mm
200mm
Inertia (pitch) = 1/12 * m * (h²+w²) + m * d
m = 3kg; **h** = 0.2m; **w** = 0.1m; **d**= $\sqrt{0.05m^2 + 0.05m^2}$
Segment of foot:
Lenth = 100mm
Vertical movement = 50mm; with 30° of rotational range
30 Nm
100 mm/s
50mm

Figure 1 Dimension assumption if foot design

With this rather conservative calculation we come to the solution that we require 60W of Motor Power. The following internet search for a suitable motors concluded we have to lower the requirements for the motor to fit it in to our budget of 300€

If we lower our torque requirement from 30 Nm to 15Nm and also changing and change our speed from 100mm/s to 50mm/s we end up with a gear moment of 2,25Nm and 63,6RPM. That is almost in the range of the following motor! (Hochleistungsgetriebemotor 12 V Modelcraft RB350050-22723R 50:1 https://www.conrad.de/de/hochleistungsgetriebemotor-12-v-modelcraft-rb350050-22723r-501-233132.html)

Design Process of the Gearbox, Toes and Heel

Gearbox includes a motor mount, a framework, 6 ball bearings, 1 bevel gears set, 3 spur gears and 2 winders. Taking into account the torque that needs to be transferred, the modules of the gears, diameter of the shafts and dimensions of the ball bearings have been calculated. Bevel gears and spur gears requires the module of at least 0,58 and 0,8, respectively.

Determining the Shaft Diameter

Torsion occurred on the shaft by DC Motor;

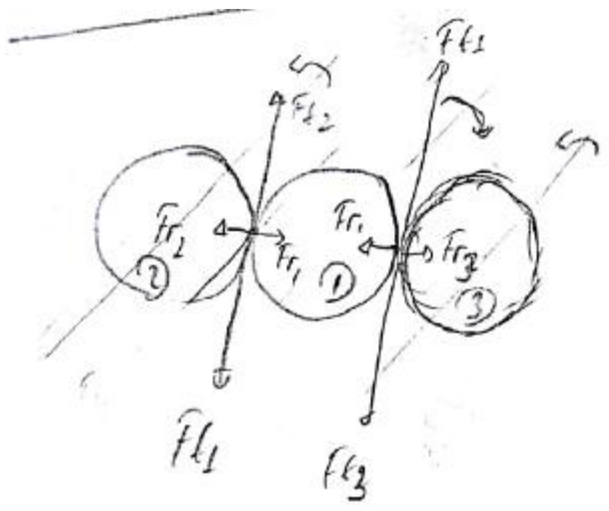
$P = 15{,}18\,W$, $n = 44{,}18\,rpm$ (taken from the DC motor catalogue considering the torque of 3,28 Nm)

$$\tau_{b_1} = \frac{16 * M_1}{\pi * d_1^3} \leq \tau_{em} = 720\,MPa$$

$$M_1 = 9550\frac{P}{n} = 3{,}28\,Nm$$

$$d_1^3 \geq \frac{16 * 3{,}28}{\pi * 720} \rightarrow d_1 \geq 2{,}85\,mm$$

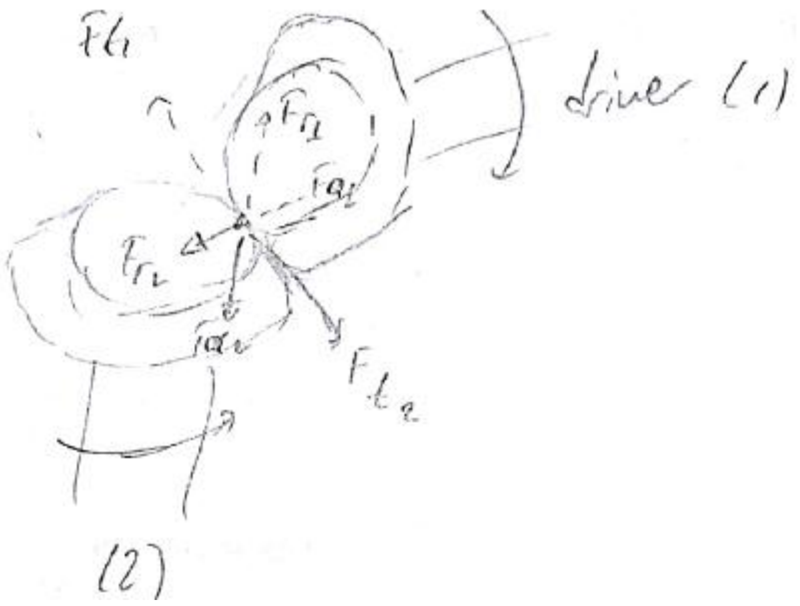
Forces acting on the spur gears;



$$F_{t_2} = F_{t_3} = \frac{2000 * T}{d} = 218{,}6\,N$$

$$F_{r_2} = F_{r_3} = F_t * \tan\alpha = 79{,}56\,N$$

Forces acting on the bevel gears;



Reference cone angles (δ_1, δ_2)= 45°, helix angle (β)= 0° ,shaft angle(∑) = 90°

pressure angle (α) = 20° , face width(b) = 7 mm , d_m= reference diameter angle

$$d_{m_1} = 31 - 7 * \sin\delta_1 = 26{,}05\, mm$$

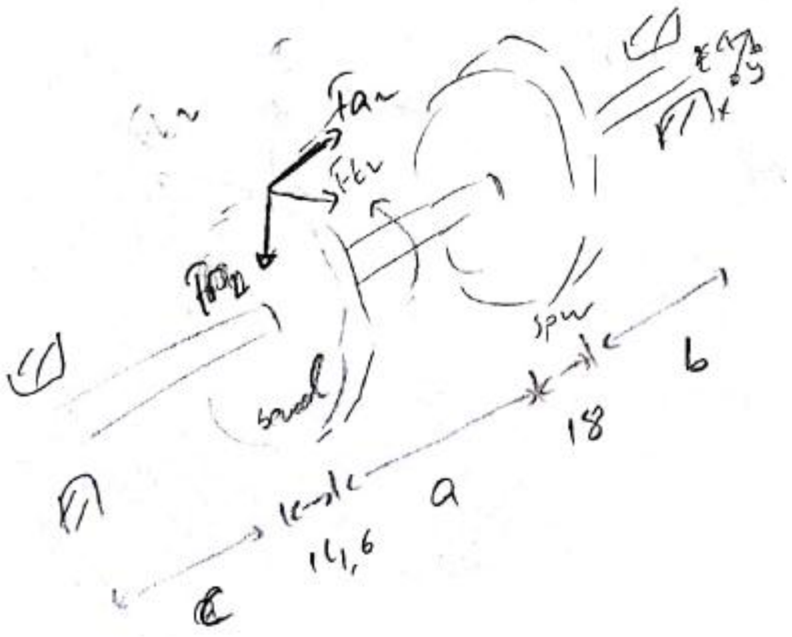
$$d_{m_2} = 31 - 7 * \sin\delta_2 = 26{,}05\, mm$$

$$F_{t_1} = F_{t_2}$$

$$F_{t_1} = \frac{2000 * T}{d_m} = 218{,}6\, N$$

$$F_{r_2} = F_{a_1} = F_t * \tan\alpha * \sin\delta = 64{,}81\, N$$

$$F_{r_1} = F_{a_2} = \ F_t * \tan\alpha * \cos\delta = 64{,}81\, N$$



$$a = 33{,}7\ mm\ , b = 16\ mm, c = 14{,}3\ mm$$

$$M_{torsion} = 3{,}28\ Nm$$

$$M_{bending} = M_x = -F_{r_2} * (a + b + 18) + F_{a_2} * 26{,}05 \approx -2{,}7\ Nm$$

According to the maximum shear stress theory;

$$M_{eq} = \sqrt[2]{M_t^2 + M_b^2} = 4{,}24\ Nm$$

$$d_1^3 \geq \frac{32 * M_{eq}}{\pi * \tau_{em}} \rightarrow d_1 \geq 3{,}91\ mm$$

6 mm has been decided as the shaft diameter considering the safety factor of 1,5.

After these calculations, we have bought gears made of Polyacetal, which meets the design criteria, silver shafts with 6 mm diameter and ball bearing with plastic cage material and balls made of stainless steel. Winders which plays the active role in the transmission was designed in the shape of a cylindrical cage so that during the assembly rope could have been easily fixed onto one of the bars regardless of the initial position of the winder. Framework, motor mount and the winders has been printed using 3D Printer.

The toes and heel parts are connected to lower adapter part in a way that lower adapter part is a bit elevated from the ground. There are 2 reasons for this setup: Main reason is to imitate the arc structure of the foot in human beings. Scientific papers show that the middle part of the foot has the lowest pressure while walking or standing and the most pressured points are the toes and heels. The second reason is due to the screws in the lower adapter part which should not contact to the ground as it would disturb the contact layer of the foot.2 torsional springs are used to imitate the bending of toes so that some of the energy can be restored and reused passively while walking.

Troubleshooting

The 3d printing process of toes and heels were a bit frustrating. At first trial 100% printing density was used but the parts were too heavy for the motor to handle. At second trial 30% is used but this time the quality of the parts was poor therefore the parts had to be reprinted many times which is very frustrating and suffering since the printing time was quite long and the student workshop was unexpectedly closed some days during the course of the semester.

Actuator - 12V DC Motor

This is the installation guide our human foot demonstrator. This part provides the installation steps for creating a DC motor control with an H-Bridge and Raspberry Pi 3 B+ and MATLAB Simulink R2018a. It covers the following:

- Used hardware part list
- Schematic of all components
- Flashing procedure of the Pi with MATLAB OS
- Network configuration for communication setup
- Manual for the motor control MATLAB script
- Manual for the motor driver in Simulink

Used hardware part list

The following list shows the used parts for this motor setup for this demonstrator.

Partname	Function
Raspberry Pi 3 B+	Microcontroller for the motor control setup
H-Bridge, Iduino L298N	DC motor controller
12V DC Motor	Actuator for the demonstrator
12V Power Supply Unit	Power supply for the DC motor
Breadboard	Used for quick wiring
Breadboard cables	Connectors for quick wiring
3x LEDs	Used for indicating the current status of the motor (turn left, turn right, stop)
Micro USB Power Supply Unit	Power cord for the microcontroller
RJ45 Cable, 1m	Network cable for connecting Pi with PC
3x resistors	Required for the lowering the input voltage of LED

Schematic of all components

The schematic of the demonstrator is straight forward. Pi 3 B+ we use the following GPIO pin layout as reference.

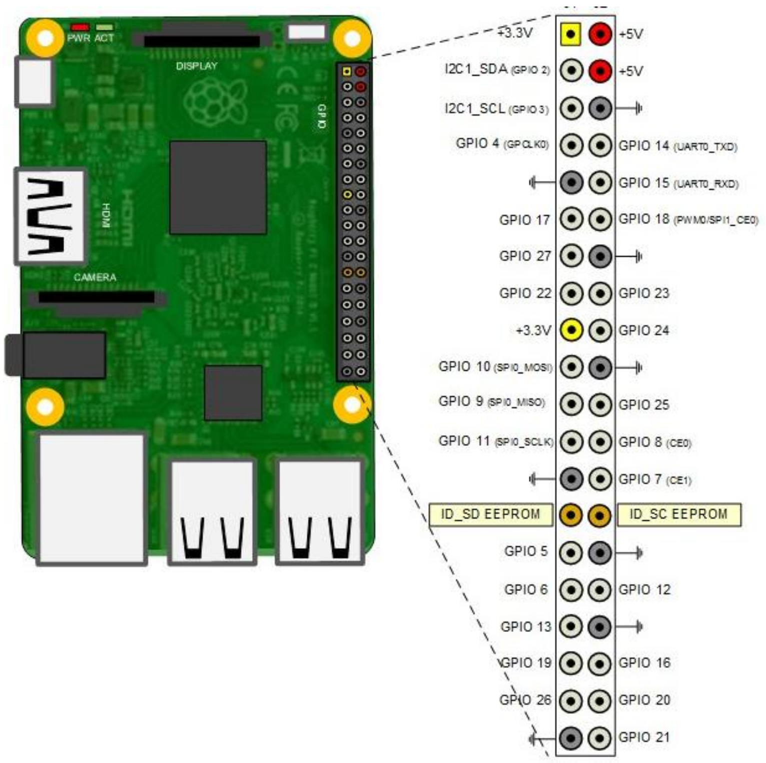


Figure 2 MATLAB showPins layout for Pi Model 3 B

From the Pi we use ground and map it on the breadboard for our LED monitoring. The LEDs need a dropping resistor otherwise the LED would break, since the output voltage of the GPIO pins are 3.3V and the permitted voltage for LEDs we have used is 1.6V. Constants:

$$U_{GPIO} = 3.3\ \mathrm{V}$$

$$I_{GPIO} = 0.016\ \mathrm{A}$$

$$U_{LED} = 2\ \mathrm{V}$$

$$R_V = \frac{U_{GPIO} - U_{LED}}{I_{GPIO}} = \frac{3.3\ \mathrm{V} - 1.6\ \mathrm{V}}{0.016\ \mathrm{A}} = 106.25\ \Omega$$

Therefore, we chose the next bigger available resistor which has a resistance of 110 Ω.

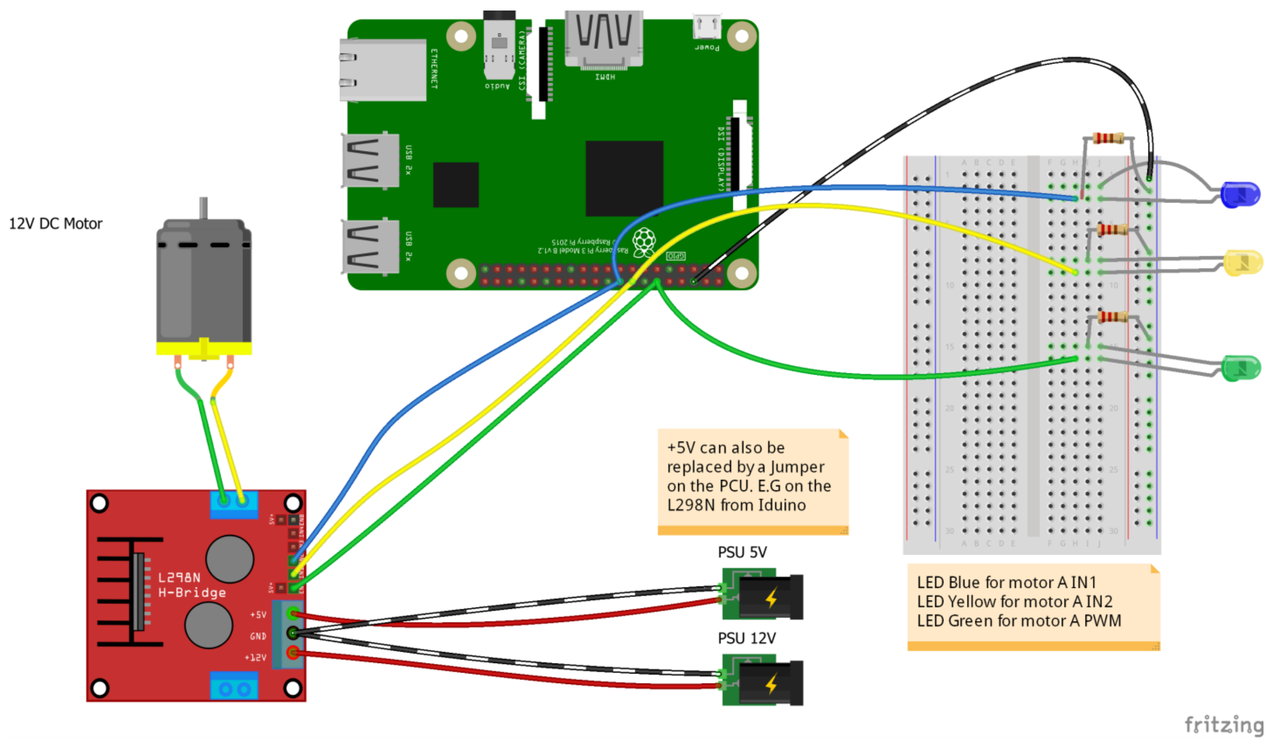


Figure 3 Fritzing Breadboard Diagramm for Motor Control

From the Pi we connect the breadboard LED control with series resistors. Use therefore ground, GPIO 18 for green, GPIO 23 for yellow and GPIO 24 blue.

Next we connect the Pi with the H-Bridge. Simply connect the GPIO 18 to Enable A (EnA control for motor A, we could control also two motors with this bridge). This allows us the control the motor speed with Pulse Width Modulation (PWM). Then connect GPIO 23 for IN2 and 24 for IN3.

In order to understand the way of controlling the motor we visualize the small motor control.

InA	InB	Motor movement
0	0	Stop
0	1	Left
1	0	Right
1	1	Stop

Table 1 Motor Control

Now we hook up the motor by using the corresponding motor pins for voltage and ground. Since we have a DC motor it doesn’t matter in with order we connect power or ground, it only affects the way of turning the motor. So be consistent and test afterwards which way the motor is turning depending on your input of InA and InB.

Lastly we need to connect power to the H-bridge. Depending on the purchased model a jumper will be shipped. The purpose of the jumper is to use 5V of the 12V power needed for the motor to supply the onboard components of the bridge with power to work properly. If you remove the jumper you need to attach a separate 5V power supply to power up the board and additional 12V for the motor power supply.

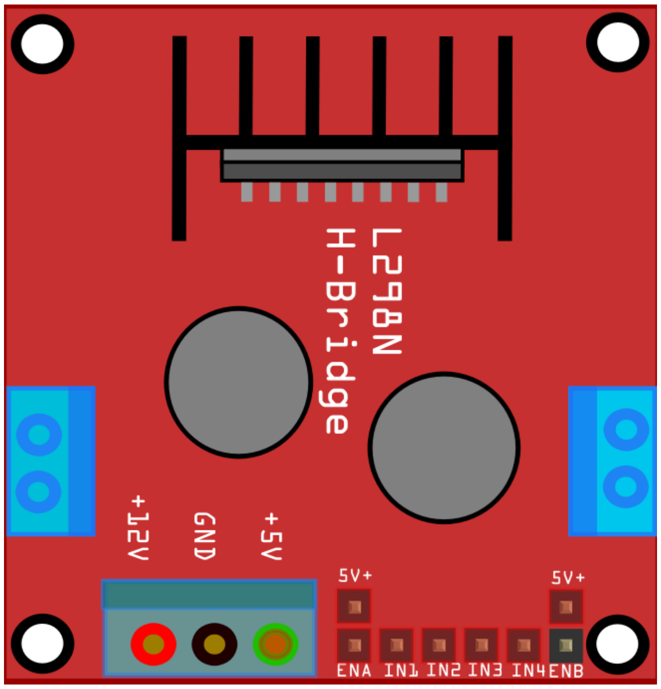


Figure 4 H-Bridge

Troubleshooting

Depending on the used power supply, place the jumper to power on the board, otherwise the board won't receive signals from the microcontroller.

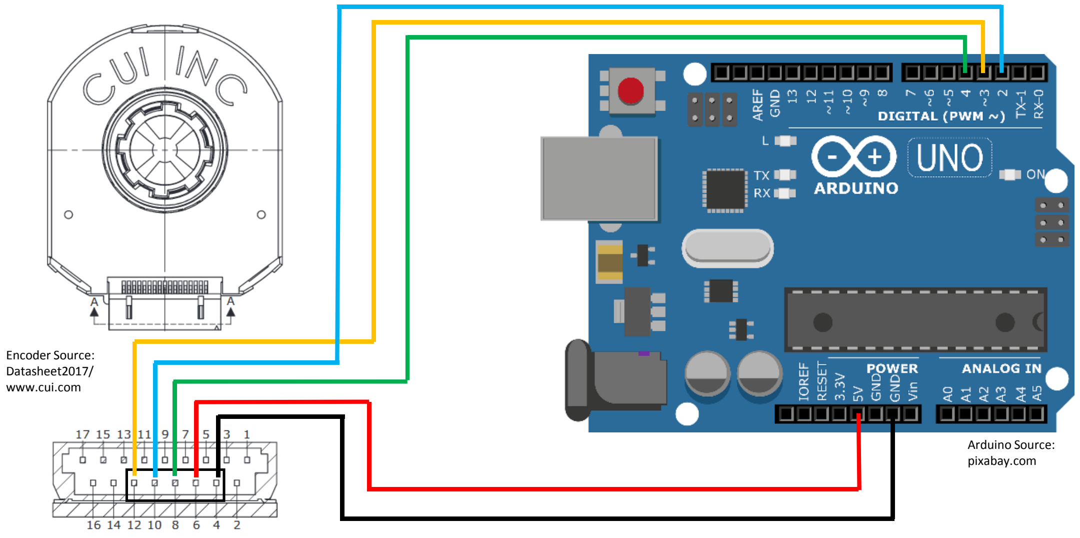
Sensors

This section gives a brief description of the implemented sensors

Encoder

The Encoder (AMT31 by CUI INC) is used to read the angle of the middle gear shaft. This corresponds (1:1) to the motor angle (behind the gear which is purchased with the motor). The connector "JAE FI-W17S" was not easily available, so we build our own, only connecting to pins 4,6,8,10,12, corresponding to GND, +5V, B+, A+, Z+.

The following Figure 5 gives the wiring and the corresponding pins



CUI INC
Encoder Source: Datasheet2017/ www.cui.com
17 15 13 11 9 7 5 3 1
16 14 12 10 8 6 4 2
AREF GND 13 12 ~11 ~10 ~9 8 7 ~6 ~5 4 ~3 2 TX→1 RX←0
DIGITAL (PWM ~)
L
TX
RX
ARDUINO
UNO
ON
IOREF RESET 3.3V 5V GND GND Vin
POWER
ANALOG IN
A0 A1 A2 A3 A4 A5
Arduino Source: pixabay.com

Figure 5 Wiring Encoder Arduino

A & B form a quadrature signal. They are the same signal but shifted in time. If A lags behind be the rotation direction is the one way (say negative), if B legs behind A it is the other direction (say positive). There is four cases when a change in A (or B) was detected A=1 & B=0; A=0 & B=1 (both positive direction) and A=0 & B=1; A=1 & B=0 (both negative direction).

Z gives a pulse whenever a zero crossing happens.

As the encoder works incremental each signal flank of either A or B must be detected to correctly sum the steps to the angular value. The encoder resolution is 4096/round (so 2048 up and 2048 down flanks). Using Simulink for Raspberry Pi we could not read the data sufficiently fast (especially when other blocks like the motor control were included) thus we moved the encoder reading to a Arduino and only send the calculated angular value via a serial connection (USB).

To always reference to the same 0 position (given by Z signal) we initially turn the motor slowly until the first Z pulse is received, only after that that initialized bit is set and the actual code may start. (Note: make sure the motor is in the correct rotation, as this cannot be detected by the encoder and initially rotation 0 is always assumed.)

For the serial connection, set the correct Port in the Simulink Block, find the value by typing

```
ls /dev/tty*
```

in the command line on the paspberry pi (e.g using putty) and see which element is added when you connect the Arduino by USB.

<u>Find in the files:</u>

Data sheet of encoder; Code for Arduino

Troubleshooting:

The encoder seemed to "pass out" at times, we are not sure for the reason, but we did not even get a signal using an oscilloscope. Possible reasons may be: Bad connector, loose connection, insufficient power supply by the Arduino, bad quality of the encoder or anything else.

Force Sensor

The force sensor (Interlink Drucksensor Force-Sensing Resistor FSR) are used to detect whether the foot is touching the ground or not. These sensors are passive elements exhibits a decrease in resistance when there is an increase in the force. The dimensions of the sensors are 18.3 x 0.46 mm(Ø x H) and have a working range from 0.2N – 20N. A typical resistance vs. force curve is shown in Figure 6 Force to Resistance and Voltage relationship of FSR; source: FSR® Integration Guide.

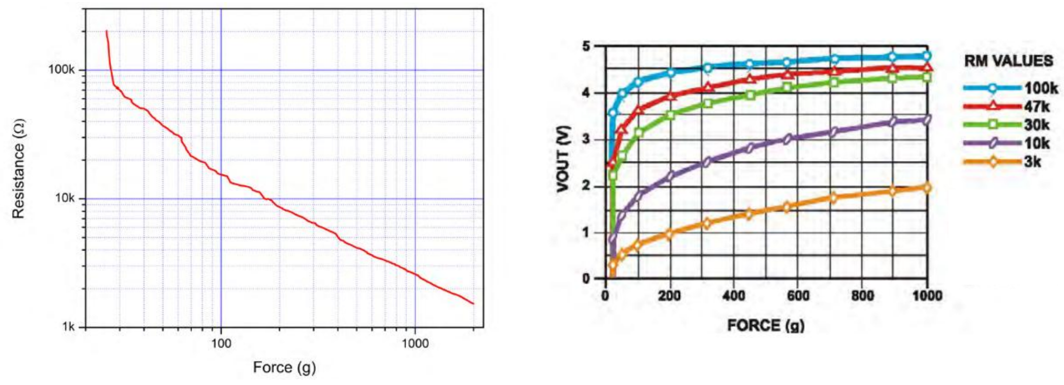
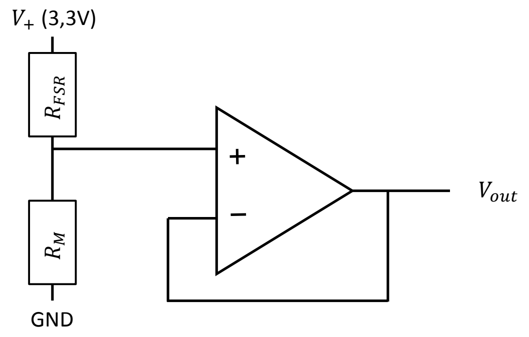


Figure 6 Force to Resistance and Voltage relationship of FSR; source: FSR® Integration Guide

The measurement setup is shown in Figure 7, a simple voltage divider circuit including the FSR Resistor is used. In the shown configuration, the output voltage increases with increasing force. The measuring resistor, RM, is chosen to maximize the desired force sensitivity range and to limit current. In our implementation RM is chosen as 57 kΩ. As amplifier the LM 393P is used, for further information check the data sheet (http://www.ti.com/lit/ds/symlink/lm393.pdf).



$$V_{out} = \frac{R_M V_+}{(R_M + R_{FSR})}$$

Figure 7

To convert the analog signal given by V_{out} to a digital signal the MCP 3008 is used. The MCP3008 device is a 10-bit Analog-to-Digital (A/D) converter with on-board sample and hold circuitry, it operates over a broad voltage range (2.7V - 5.5V). Our operation range is the 3.3V output from the Arduino Uno R3. The MCP 3008 has 8 input channel, we only need 4 channel, the wiring to the Arduino is shown in Figure 8. The communication is done via the spi interface. To read out channel 0-3 we followed the tutorial from Robert Fromm (http://www.robert-fromm.info/?post=elec_spi_mcp3008), also used his code.

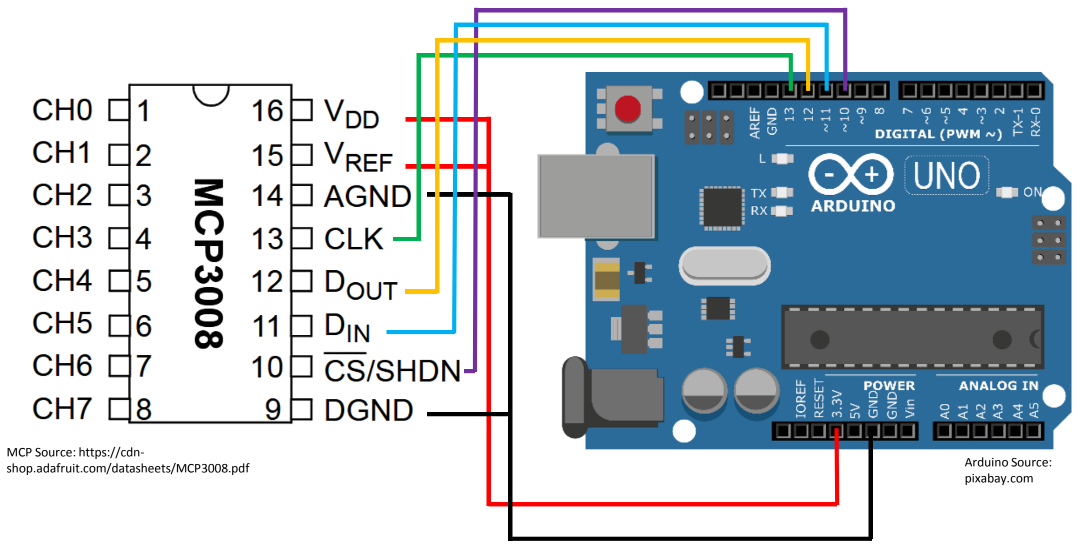

CH0
CH1
CH2
CH3
CH4
CH5
CH6
CH7
MCP3008
16 VDD
15 VREF
14 AGND
13 CLK
12 DOUT
11 DIN
10 CS/SHDN
9 DGND
MCP Source: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf
DIGITAL (PWM ~)
ARDUINO
UNO
POWER
ANALOG IN
IOREF RESET 3.3V 5V GND GND Vin
A0 A1 A2 A3 A4 A5
AREF GND 13 12 ~11 ~10 ~9 8 7 ~6 ~5 4 ~3 2 TX-1 RX-0
Arduino Source: pixabay.com

Figure 8 MCP3008 to Arduino wiring

Troubleshooting: To detect the ground contact more reliable a small paper was placed under the Force sensors

IMU

Reason to use IMUs: For the control loop it is necessary to calculate the pitch and roll. This can obviously be achieved using encoders, but they are quite expensive. Therefore we wanted to try out the cheaper IMUs. They are used to calculate the roll pitch (yaw) between the earth frame and the sensor frame. This can in turn be used to calculate the relative angle between them.

For 1 DOF joints this is achievable with only 6 axis IMUs (gyroscope+accelerometer) assuming the sensor axis allign with the axis of rotation (or the transformation is known). For a more general and theoretically more robust method we decided on 9 axis IMU (also includes an magnometer).

Choice of Hardware: We decided to use the MPU9250 from Invensense due to its cheap price and its frequent use in the open source community, therefore offering a good basis in case of problems to fall back on. In the sources folder you find the product information, register map as well the orientation of the MPU.

Note that we didnt buy the MPU9250 directly but rather bought a breakout board for its ease of use.

To run the sensor fusion we orginally wanted to use a Raspberry Pi but run into some problems with simulink and first turned to the Arduino UNO. While we got the fusion (somewhat, see below) working for one sensor, we figured the UNO is still to slow to offer results in the acceptable range. Hence we used a Teensy 3.6. Its more than ten times faster than an UNO and offers 4 I2Cs Channels/Ports/Pins.

Senor Fusion: As already remarked we orginally thought of the using the Raspberry Pi. This would have probably resulted in quite a low sampling rate. Hence we looked for a sensor fusion still offering acceptable results at low rates who would also be computationally inexpensive. The fusion algorithm designed by Magdwick fits the pill (paper in sources, http://x-io.co.uk/open-source-imu-and-ahrs-algorithms/).

Calibration: In theory all individual sensors on the MPU9250 (acc, gyro, mag) have to be calibrated. In practise we have only calibrated the offset in the gyroscope(absolutly necessary) and the scaling error as well offset magnometer (absolutly necessary). For the first we simply took x amounts of samples while the gyros werent moved and averaged it to calculate the offset. The scaling error couldnt be calibrated, because we were lacking a reference angluar velocity. The magnometer calibration is very important. There exist quite a few ways to this. We went with a simple mapping as described here (https://github.com/kriswiner/MPU6050/wiki/Simple-and-Effective-Magnetometer-Calibration). Most simple magnometer calibration rely on mapping ellipsoid measurements to a sphere (for one see sources). At this point its important to note that our magnometer wasnt well calibrated and we experienced poor performance due to it. So you would be well advised to look for other calibration methods.

Set Up: The Teensy is connected to the Raspberry/PC over a micro USB cable. The sensor are connected to the 3.3V and Ground of the teensy. Each indiviual IMU is connected to an I2c channel of the teensy. Furthermore 4.7k pull up resistors have to used (This is a must, otherwise you wont get readings).

Code: The code was written for the teensy but its close to known Arudino libaries. Its recommended to know some C++. Furthermore you will have to install teensyduino.

Performance: We had good results for the roll and pitch but pretty bad ones for the yaw. We believe this is due to the insufficient calibration.

Troubleshooting:

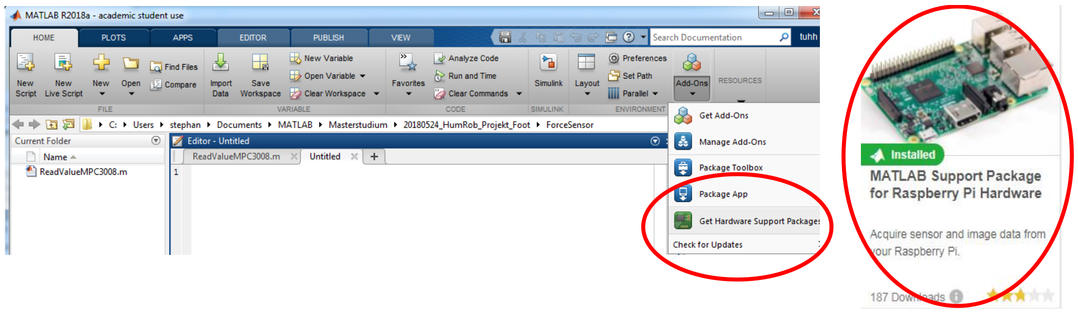
Matlab/Simulink

Flashing procedure of the Pi with MATLAB OS

There are two options to flash the SD card with the raspbian operating system including all packages from matlab.

Option 1 – Tutorial via Matlab

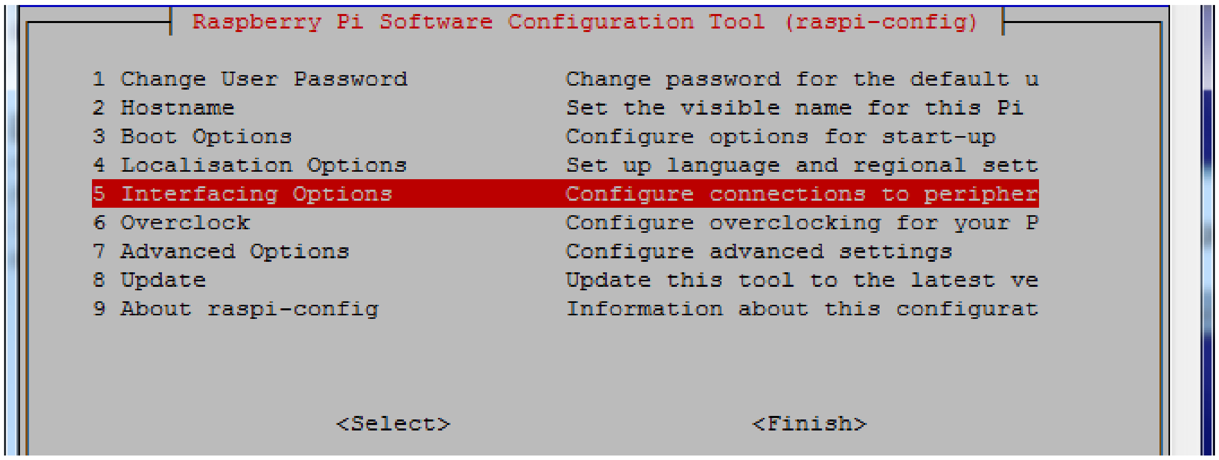
There exists a tutorial provided by Matlab to set up your raspberry pi. If this works *great!* otherwise turn to option 2.


MATLAB R2018a - academic student use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Add-Ons
Get Add-Ons
Manage Add-Ons
Package Toolbox
Package App
Get Hardware Support Packages
Check for Updates
Installed
MATLAB Support Package for Raspberry Pi Hardware
Acquire sensor and image data from your Raspberry Pi.
187 Downloads

Note: The raspbian op provided by matlab was not bootable with the raspberry pi model 3b+

Option 2 – Manual installation

Firstly download the newest version of raspbian op (https://www.raspberrypi.org/downloads/raspbian/), this should guarantee to work with your raspberry pi. Afterwards flash your SD card with the downloaded op (*Hint: We used Win32 Disk Imager*). Insert the SD Card in your pi and boot the op system. Make sure you have installed all updates, therefore enter **sudo raspi-config** into the console.



```
        Raspberry Pi Software Configuration Tool (raspi-config)

   1 Change User Password         Change password for the default u
   2 Hostname                     Set the visible name for this Pi
   3 Boot Options                 Configure options for start-up
   4 Localisation Options         Set up language and regional sett
   5 Interfacing Options          Configure connections to peripher
   6 Overclock                    Configure overclocking for your P
   7 Advanced Options             Configure advanced settings
   8 Update                       Update this tool to the latest ve
   9 About raspi-config           Information about this configurat


                  <Select>                     <Finish>
```

Abbildung 1 Source: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c

Under 8.Advanced Options the raspberry pi can be updated. Afterwards reboot and install the missing matlab packages with the following command.

sudo apt-get install matlab-rpi

If the installation was successful run **sudo raspi-config** and select 5.Interfacing Options, enable ssh, i2c and gpio connection. Afterwards reboot your pi and connect it to your pc via a lan-cable. Make sure you know the IPaddress of your pi. Run **sudo ifconfig** to find the IP address next to the wlan0 entry. At last run the **rasp('IP-Adress','user name','password')** command in matlab.

To verify the installation reinstall the Matlab Support Package for Raspberry Pi Hardware witch Matlab on an existing operating system.

Troubleshooting

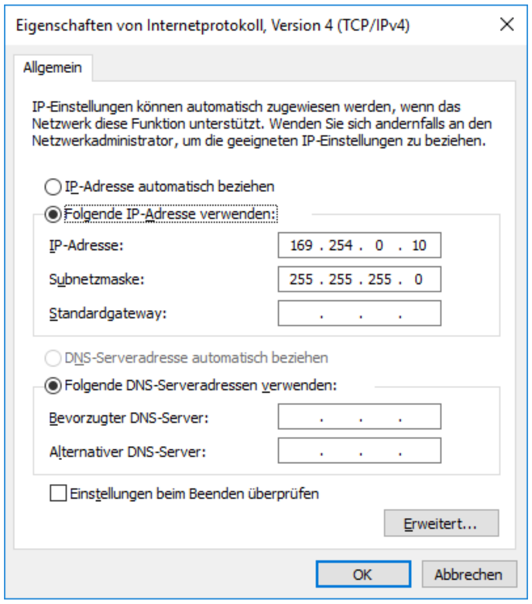
*If the installation of the matlab packages was not possible in one step there is the option to install the packages one by one. Also note you have to run the command **rasp('IP-Adress','user name','password')** at least once before you can verify your installation with the "Matlab Support Package for Raspberry Pi Hardware".*

One by one installation: run the following command and replace the xxx with the listed package names

sudo apt-get install xxx

libsdl1.2-dev	alsa-utils	espeak	i2c-tools
libi2c-dev	ssmtp	ntpdate	git-core
v4l-utils	cmake	sense-hat	sox
libsox-fmt-all	libsox-dev	libcurl4-openssl-dev	

Network configuration for communication setup


Eigenschaften von Internetprotokoll, Version 4 (TCP/IPv4)

Allgemein

IP-Einstellungen können automatisch zugewiesen werden, wenn das Netzwerk diese Funktion unterstützt. Wenden Sie sich andernfalls an den Netzwerkadministrator, um die geeigneten IP-Einstellungen zu beziehen.

IP-Adresse automatisch beziehen [✓ Folgende IP-Adresse verwenden:]

IP-Adresse: 169 . 254 . 0 . 10
Subnetzmaske: 255 . 255 . 255 . 0
Standardgateway: . . .

DNS-Serveradresse automatisch beziehen [✓ Folgende DNS-Serveradressen verwenden:]

Bevorzugter DNS-Server: . . .
Alternativer DNS-Server: . . .

☐ Einstellungen beim Beenden überprüfen

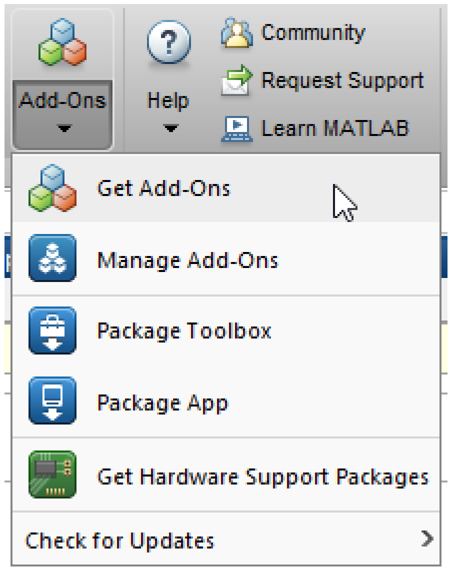
During the flashing of the raspberry Pi one was asked to define a proper IP address to the PI. We have chosen: 169.254.0.2.

After that we need to hook up the PI with our PC using the Ethernet cable. Then we need to re-configure our network IPv4 address to: 169.254.0.10 / 255.255.0.0

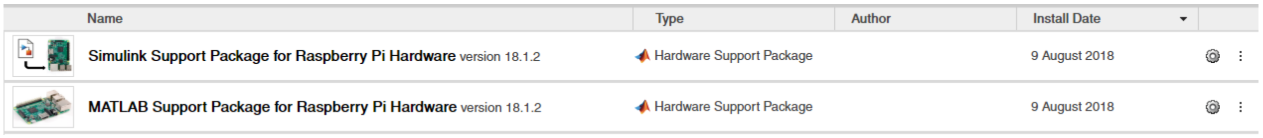
Once this is done we can start Matlab to run our setup code.

Manual for the motor control MATLAB script

As a pre-reqiusited for using matlab on raspberry pi you need to install the current support packages for matlab and Simulink, which can be easily found via Get Hardware Support Packages.



Then install Simulink support package for raspberry pi hardware and matlab support package for raspberry pi hardware.



Name	Type	Author	Install Date
Simulink Support Package for Raspberry Pi Hardware version 18.1.2	Hardware Support Package		9 August 2018
MATLAB Support Package for Raspberry Pi Hardware version 18.1.2	Hardware Support Package		9 August 2018

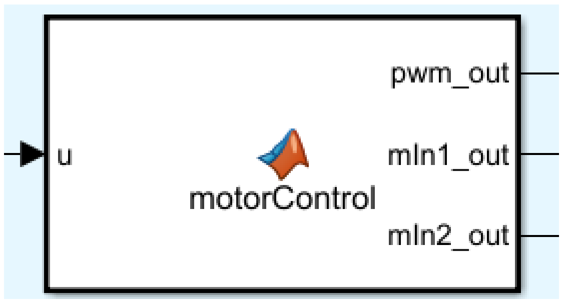
Via installation follow the instructions of the install wizard and you good to go.

For the motor control we have set up a dedicated matlab script for initial testing. Within there we have placed additional comments for easy understanding of the functions from the MATLAB raspberry PI API.

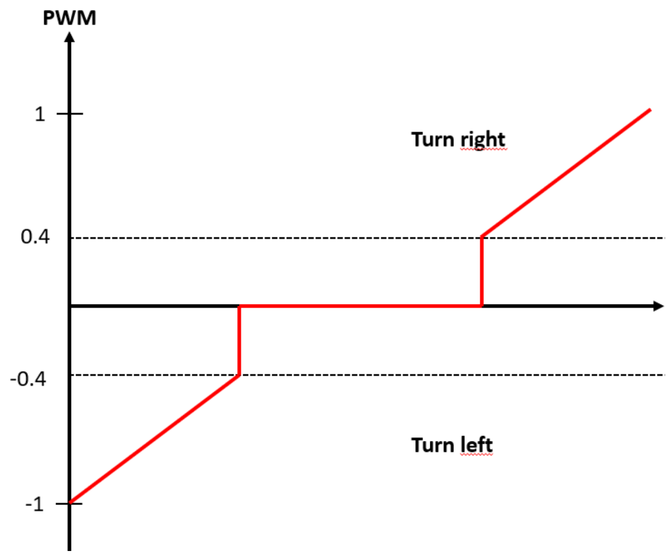
An overview of the function used can be found on following link:
https://de.mathworks.com/help/supportpkg/raspberrypiio/functionlist.html

Manual for the motor driver in Simulink

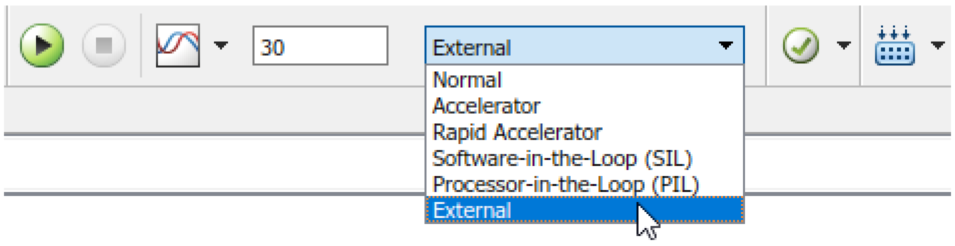
Within Simulink we have created a motor control block which takes and input u which can be between [-1,1]. This input should come from another system block which provides new directives turning or stopping the motor. Output signals are pwm_out (for motor speed), mIn1_ and ,In2_out for the turning and stopping the motor.



The control scheme looks as follows:



Before running the simulation check first if the simulation is selected to external. Otherwise you won't be able to see anything in your scope when you would like to check some signals.

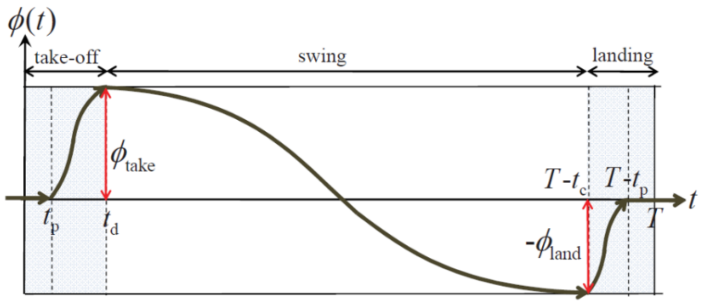


Reference Signal for the ankle

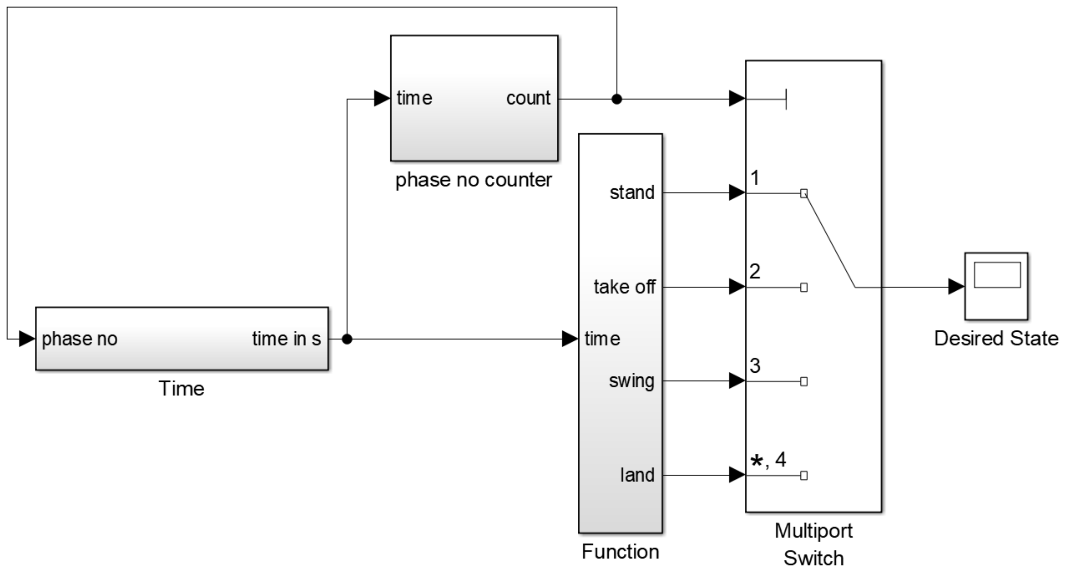
According to Kwons and Parks paper "Kinesiology-based robot foot design for human-like walking." from 2012, the foot movement happens in four phases:

1. Standing (PHI = 0, foot has full ground contact)
2. Take-off (PHI > 0, PHI increases, foot has some ground contact)
3. Swing (PHI decreases, foot is in the air)
4. Landing (PHI < 0, PHI increases, foot has some ground contact)

(This numbering conforms the one in Simulink.)



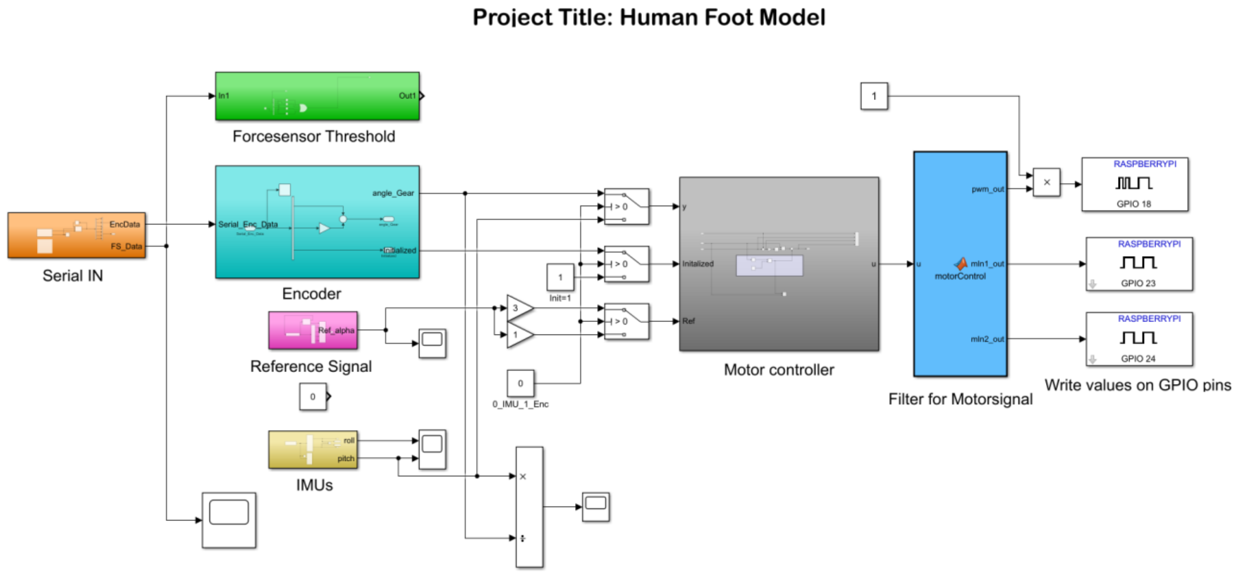
To be able to start off with slower movements, a scaling factor is used. If it's set to one, T equals the suggested value of 2 seconds. The Simulink model which provides this curve looks like this:



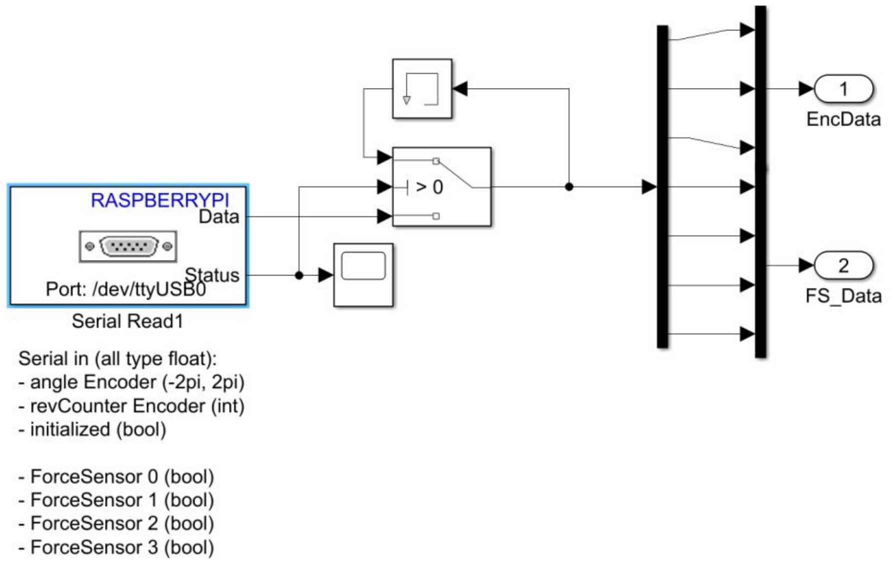
The Time-Subsystem counts the time of a phase in seconds and provides it as a variable for the Function Subsystem, where the functions of all 4 phases are being calculated. The multiport switch then picks the right function based on the phase which is actually happening. This information comes from the phase no counter subsystem. For more insights look into the documentation of this.

Overall Control Model

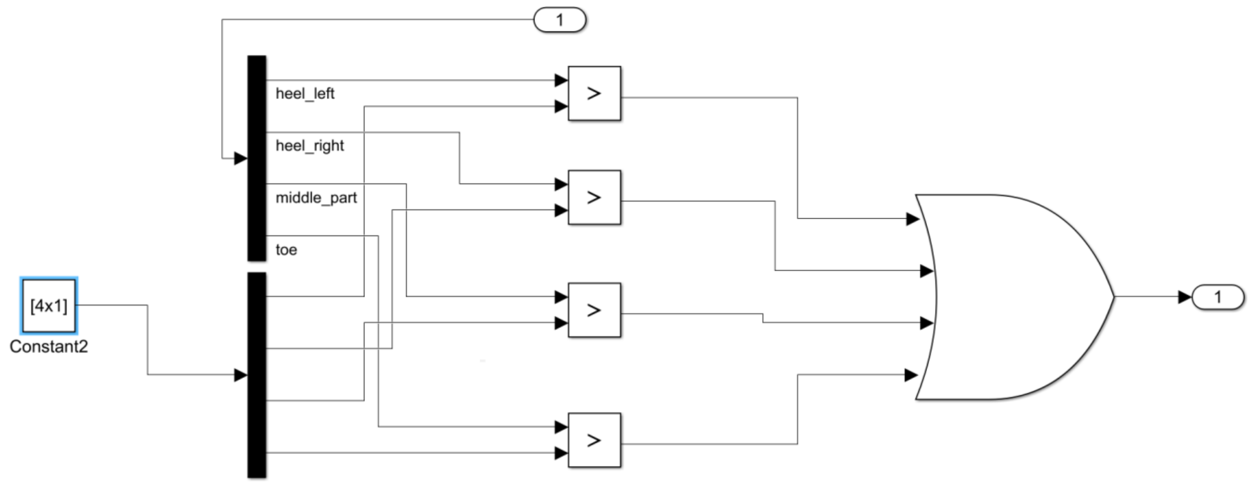
In the following segment some general remarks for the Simulink model for the demonstrator will be made. The model will be introduced in the following graphic.



The model contains blocks to read out data from the different sensors, control the motor and to write the values of the motor controller onto the GPIO pins of the raspberry pi. It starts with a “Serial IN” block which is used to read out the data from the force sensors and the encoder (which are both preprocessed on the Arduino) over the USB port of the raspberry pi (connected to the Arduino). The block diagram looks as followed.

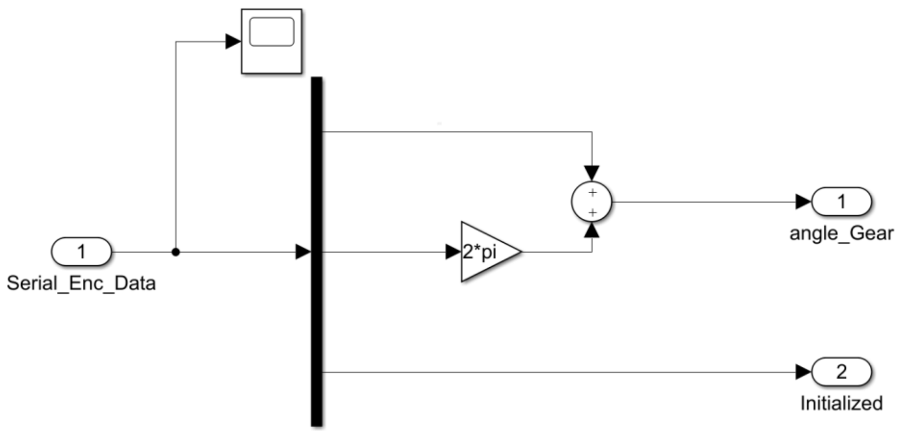


The “Serial IN” block is followed by the “Force sensor Threshold” and the “Encoder” blocks which are used to process the acquired data. The force sensors are so far only used to determine whether the foot touches the ground or not which is used to trigger the movement of the foot.



Here the input vector from the sensors gets demuxed and compared to a distinct value which is one since we are only detecting whether the foot is in contact with the ground or not. Therefore, if one of the sensors give a value greater than 1, the foot is in contact with the ground and the output of the block will be a logical 1 which is supposed to trigger the foot motion in future projects.

The “Serial IN” also delivers an input for the encoder block, which is illustrated in the following picture.

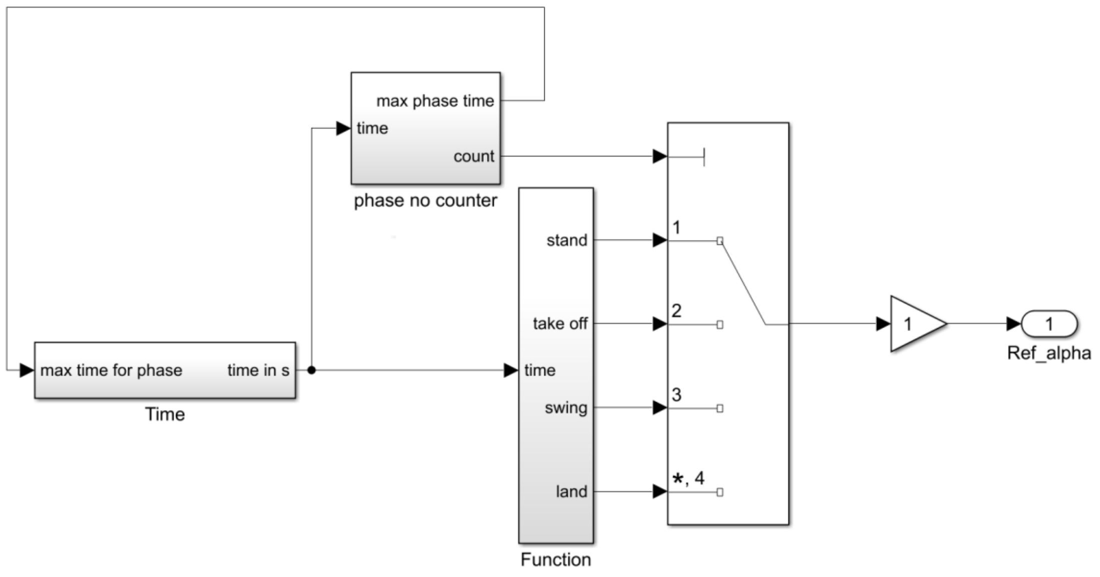


Here the collected data (see the previous Encoder section) will be demuxed. The data vector contains the following information:

1. The angle of the foot in an interval of (-2pi; 2pi)
2. The full revolutions counted (using the encoders Z signal)
3. A flag indicating if the encoder is initialized (i.e. at least one zero crossing was registered before)

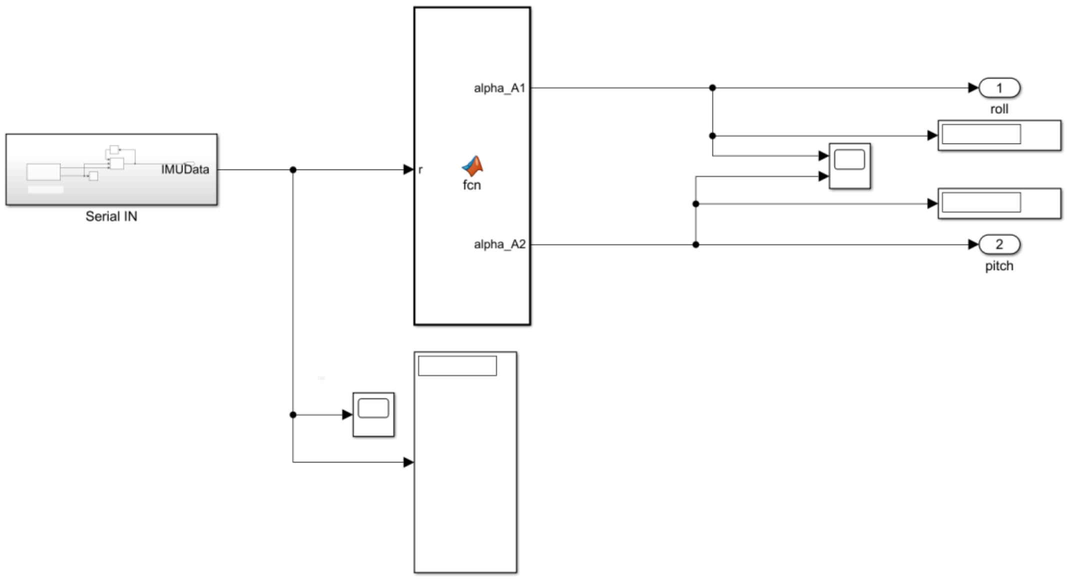
The full angle is calculated by adding the current angle to the 2pi multiple of the revolution counter.

The reference signal block is described in the previous subchapter and additionally documented in the code.

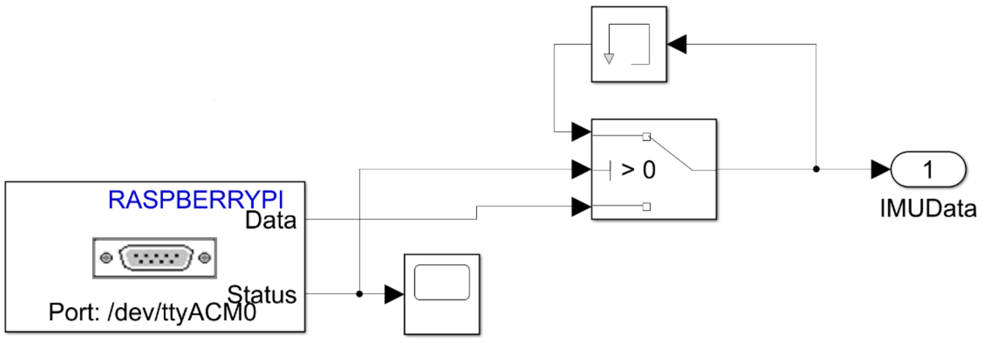


The reference angle has to be scaled before it gets fed into the motor controller.

The IMU block looks as followed.



It has its own “Serial IN” block which is used analogously to the previous “Signal IN” block to read out the data from the IMUs, which are preprocessed by the MPU9250.

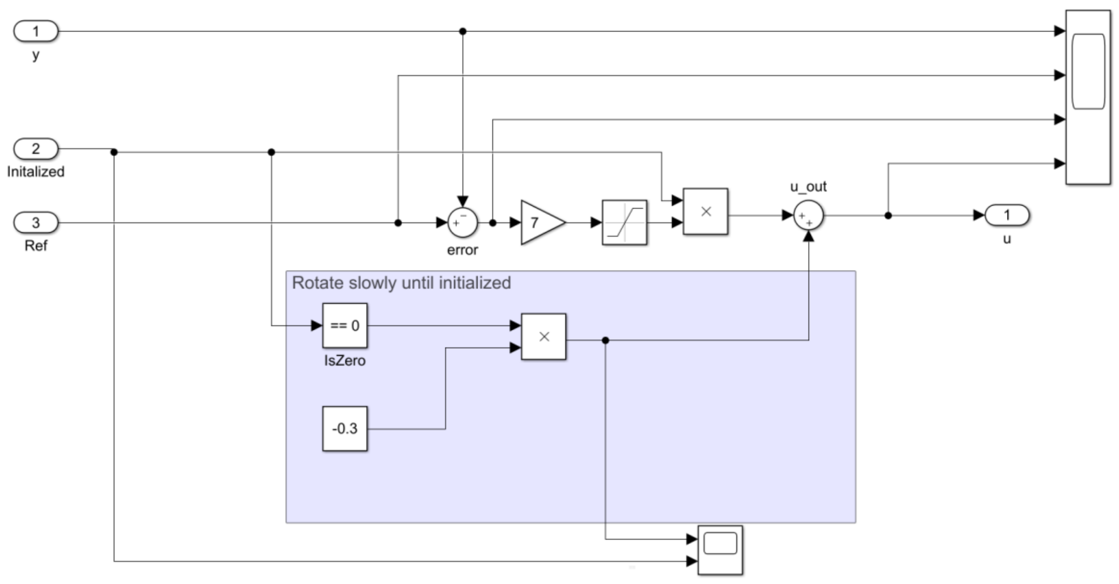


Serial in (all type float):
- roll pitch yaw for all 3 IMUS

The input signal is processed in a function block, which outputs two angles (pitch and roll) for the motor controller.

Currently only one of the reference signals (either encoder or IMU angles) is used for control. A later state space model may use both simultaneously

The motor controller uses the processed sensor data of one of the two sensors and the reference signal as its input. In case the encoder is being used, the controller starts performing a slow initialization movement unless it’s initialized already. The control scheme which is implemented at the moment is a classical P-Controller, which compares the sensor signal with the reference. It will then feed the output values to the motor filter, which is described in the “Manual for the motor driver in Simulink section”.



We used the following P values (sign depends on the direction in which the motor is mounted relative to the joints):

- Encoder based control P=-1
- IMU based control P = 5…7

After the signal got filtered the values get written on the GPIO pins of the raspberry pi which then actuates the foot.

Outlook

Design

- Gears will jump if too much load is on the string which is connected on the barrell, use springs to preload the strings
- If you use 3d printer for parts use structure density of 30% due to reduced part weight
- Reduce the wooden shaft length in order to reduce weight and size of leg
- Build an external construction to lift the leg off the ground, otherwise one person needs to hold the leg up in the air which can be very exhausting

Microcontroller

- Do not use raspberry pi for the simulation. the matlab and simulink packages are not properly supported from mathworks, use arduino instead.

Force Sensor

- Filter Signal (e.g. low pass)

IMU

- Better calibration. Especially for the magnometer
- Switch to SPI, if i2c is bottlenecking your speed
- Try a different sensor fusion algorithm. Magdwick is fine if you only have limited processor speeds. The teensy offers more than enough therefore it might be sensible to switch to a more computational intensive fusion who provides higher accuracy

Reference Signal

- Add Triggers & Feedback from other leg to make real walking possible

Sole

- Material was not suitable for a good fixing on 3D-Printed parts. Use different material

Appendix

Shopping list

Raspberry PI 3 B+ All-In-Bundle	https://www.reichelt.de/Einplatinen-Computer/RASP-3-B-ALL-IN/3/index.html?ACTION=3&LA=2&ARTICLE=222811&GROUPID=8242&artnr=RASP+3+B%2B+ALL+IN&trstct=pol_0
jumper cable	https://www.reichelt.de/Entwicklerboard-Zubehoer/DEBO-KABELSET/3/index.html?ACTION=3&LA=517&ARTICLE=161046&GROUPID=8244&trstct=lsbght_sldr::217696
breadboard	https://www.reichelt.de/Experimentier-Steckboards/STECKBOARD-1K2V/3/index.html?ACTION=3&LA=517&ARTICLE=67678&GROUPID=7791&trstct=lsbght_sldr::161046
Lan cable	https://www.reichelt.de/Patchkabel-Netzwerkkabel-Cat-5-e-/GOOBAY-73072/3/index.html?ACTION=3&LA=2&ARTICLE=197640&GROUPID=5847&artnr=GOOBAY+73072&trstct=pol_6
DC-Motor	https://www.conrad.de/de/hochleistungsgetriebemotor-12-v-modelcraft-rb350050-22723r-501-233132.html
Motor Controller (L298N)	https://www.conrad.de/de/motortreiber-motor-driver-board-1525437.html
Bevel gear	https://www.conrad.de/de/polyacetal-kegelrad-reely-modul-typ-10-anzahl-zaehne-30-30-1-pckg-1515602.html
Spur Gear	https://www.conrad.de/de/polyacetal-stirnzahnrad-reely-modul-typ-10-bohrungs-o-6-mm-anzahl-zaehne-30-1515593.html
Silver steel shaft	https://www.conrad.de/de/silberstahl-welle-reely-o-x-l-6-mm-x-500-mm-237086.html
A/D Converter	https://www.conrad.de/de/datenerfassungs-ic-analog-digital-wandler-adc-microchip-technology-mcp3008-ip-extern-pdip-16-651456.html
rubber mat	https://www.conrad.de/de/anti-rutsch-matten-l-x-b-x-h-20-cm-x-10-cm-x-8-mm-petex-43710104-regupol-7210-ls-plus-uv-lichtbestaendig-natriumchlor-1594087.html
IMU	https://www.exp-tech.de/sensoren/beschleunigung/7511/sparkfun-imu-breakout-mpu-9250
Encoder	https://www.mouser.de/ProductDetail/CUI/AMT312S-V?qs=sGAEpiMZZMsWp46O%252bq11WX0VmCXIU00qD2iPTVNV8yQ%3d
Interlink force sensor	https://eckstein-shop.de/Interlink-Drucksensor-Force-Sensing-Resistor-FSR-402-02-N-20-N-183-x-046-mmR-x-H
Spring tilt	https://www.febrotec.de/content.php?seite=shop/produkte.php&details=16671&hauptrubrik=23
Spring roll	https://www.febrotec.de/content.php?seite=shop/produkte.php&details=16226&hauptrubrik=23

spring toe	https://www.federnshop.com/en/products/torsion_springs/t-19206l.html

Arduino Code

```
// include the library code:
#include <LiquidCrystal.h>
#include <SPI.h>

typedef union
{
  float number;
  uint8_t bytes[4];
} FLOATUNION_t;

const uint8_t CS = 10; // Cock PIN SPI
const double pi = 3.14159;

// PINs Encoder
const int pinA = 2;
const int pinB = 4;
const int pinZ = 3;
const int resolution = 4096/(2*pi);

// dynamic Values
double angle = 0;    //[-2pi, 2pi]
double revCounter = 0;
bool initialized = false;
bool toogle = false;

// Serial Communication
void send2Simulink(float tt) {
  FLOATUNION_t fa;
  fa.number = tt;
  for (int i = 0; i<4; i++)
  {
    Serial.write(fa.bytes[i]);
  }
}

// Called at every resolution step
void change_A(){
  if (initialized)
  {
    bool A = digitalRead(pinA);
    bool B = digitalRead(pinB);

    if ((A && !B ) || (!A && B)) {
      angle += 1.0/resolution;
    }else{
      angle -= 1.0/resolution;
    }
  }

  if (toogle){
    digitalWrite(LED_BUILTIN, HIGH);
  }else{
    digitalWrite(LED_BUILTIN, LOW);
  }
  toogle = !toogle;
}

// Called whenever full 2pi are passed
void rise_Z(){
  if (angle > pi){
    revCounter +=1;
  }else if(angle < - pi){
    revCounter -= 1;
  }else{
    //Do nothing as no full revolution assumed
  }

  // Reset Angle to zero
  angle = 0.0;

  initialized = true;

}

// From http://www.robert-fromm.info/?post=elec_spi_mcp3008
uint16_t mcp3008_read(uint8_t channel) {
  digitalWrite(CS, LOW);
  SPI.transfer(0x01);
  uint8_t msb = SPI.transfer(0x80 + (channel << 4));
  uint8_t lsb = SPI.transfer(0x00);
  digitalWrite(CS, HIGH);
  return ((msb & 0x03) << 8) + lsb;
}

float forceSensor_Evaluator(uint16_t value){

  return (float)value;
//  uint16_t threshold = 80;
//
//  if (value < threshold){
//    return 0.0;
//  }else{
//    return 1.0;
//  }
}

void setup() {
  //Serial Communication
  Serial.begin(230400);
  // SPI ################
  SPI.begin();
  SPI.setClockDivider(SPI_CLOCK_DIV8);

  pinMode(CS, OUTPUT);
  digitalWrite(CS, HIGH);

  // #### ENCODER ######################################
  pinMode(pinA, INPUT);
  pinMode(pinB, INPUT);
  pinMode(pinZ, INPUT);

  attachInterrupt(digitalPinToInterrupt(pinA), change_A, CHANGE);
  attachInterrupt(digitalPinToInterrupt(pinZ), rise_Z, RISING);

   pinMode(LED_BUILTIN, OUTPUT);

   digitalWrite(LED_BUILTIN, HIGH);
}

void loop() {
   // SPI Force Sensor
  float reading_F0 = forceSensor_Evaluator(mcp3008_read(0));
  float reading_F1 = forceSensor_Evaluator(mcp3008_read(1));
  float reading_F2 = forceSensor_Evaluator(mcp3008_read(2));
  float reading_F3 = forceSensor_Evaluator(mcp3008_read(3));

  // Encoder SEND
  send2Simulink(angle);
  send2Simulink(revCounter);
  send2Simulink(initialized);

  // FORCE Sensor Send
  send2Simulink(reading_F0);
  send2Simulink(reading_F1);
  send2Simulink(reading_F2);
  send2Simulink(reading_F3);

  delay(100);

}
```