Graduation Report

Name of Student:	Colin Schipper
Subject Title:	Remote controlled robotic arm movement

Abstract

Research was conducted to enable remote control of a humanoid robotic arm using movements of the human arm. Several options for the measurement technique have been discussed, resulting in a proposal for usage of MARG sensor modules to measure human arm movement. The InvenSense MPU9250 sensors where used, since these had the best performance at the price. The AHRS algorithm proposed by Madgwick's research has been used to increase the MARG sensor data accuracy. Furthermore, an angle conversion method has been proposed, to generate angles for the robot arm based on the MARG sensor data. The sensors where calibrated and tested in cooperation with the filter on small cables and the angle conversion algorithm had been tested for its calculations in MATLAB. It was not possible to get the proof of concept working within this research, but the separate parts show that the proposes system is a viable option for remote control of a humanoid robotic arm.

Contents

List of figures

List of tables

Rationale

There are times the environment is dangerous or hazardous for humans. An example of such an environment are the liquid animal manure pits on farms. Such pits need to be cleaned regularly, however the fumes in these pits (especially hydrogen sulphide) are toxic to animals and humans. These environments can cause sickness or even death to a human, while electronic parts can often sustain the problems better since these parts can be protected better. In these situations, remote controlled robots can be used. This ends the necessity of humans to enter this environment.

Sogeti has been working on a humanoid modelled robot starting from the torso (torso, arms and head), called SPERA (**S**ogeti **P**ersonal **E**lectronic **R**obotic **A**ssistant). SPERA is mainly used for research purposes since this is built as an innovation project, for employees to work on while they are not working on a commercial project. The results of the project are used to increase the knowledge of the employees and as information for the clients of Sogeti. The arms of SPERA are currently able to move based on models, on end-effector positions or on a recorded motion by manually moving the arm in the desired position. A new task for the robot is to be remotely controlled. On SPERA this type of movement is used for demonstrative purposes, however such a system could reduce the necessity for humans to enter dangerous environments, since the operator can be far away from the robot.

Figure 1: Sogeti Personal Electronic Robotic Assistant (SPERA).

A basic set of requirements from the client can be found in Table 1. From this list, the ability to work in several lighting conditions and the time needed to learn how to correctly operate a system to control the robot arm are the most important for a final product, since this determines the ability to use the system for many operators and the possibility to deploy the system in different environments.

Table 1: client requirements for remote control of SPERA

number	Description of requirement
SF1	The system shall enable to robot to be remote controlled
SF1.1	The system shall produce the parameters needed for movement of the robot arm and gripper
SF1.2	The system shall give accurate movement in all lighting conditions
SF2	The system shall be battery powered
SF3	The system shall communicate wirelessly with the robot
NF1	A new user should be able to operate the system in a short time

SF = functional requirement, NF = non-functional requirement

Possibilities for a remote control of the robot arm are using a single or multiple magnetic angular rate ang gravity (MARG's) sensors spread over the body of a human, joystick controllers that are operated by humans or robot movement based on image recognition on a moving human arm.

Research will be conducted to gather insights into several aspects of a sensor system to remotely control a humanoid robot arm, based on the following research question. What measurement technique will enable a humanoid robot arm to be accurately controlled from remote locations. The sub-questions are: what sensor type currently gives data for the robot arm with the closest resemblance to a human arm; what sensor data

accuracy is needed for accurate movement of the robot arm; what conversion is needed to transform the sensor data into the angles that the robot arm needs for operation. This research will be done based on the available literature combined with a proof of concept to test the proposed system.

Situational & Theoretical analysis

image based movement recognition

The use of single camera and stereo camera approaches used to be common approaches in image-based gesture recognition. Single camera recognition can be done with most generic cameras that have an appropriate sample rate and resolution. The approach has limits within view angles, which affect system robustness and usability. Stereo camera uses multiple cameras to make a 3d approach to the environment in which the cameras are placed, which is currently still computationally complex and causes difficulties with calibration of the systems. Depth sensing technologies have emerged rapidly in the last few years. A depth sensor is defined as a non-stereo depth sensing device. These depth sensors have several advantages compared to more traditional cameras, since the drawbacks of the setup calibration and illumination conditions have been reduced. The output of a depth sensor is 3d depth information, which simplifies the gesture identification when compared to colour information. [1]

Recognition of objects or gestures begins with detection and segmentation of the object. Segmentation is crucial as it isolates the task-relevant data from the background of an image, before the usage in tracking and recognition. Segmentation of colour has been utilized before using a selection within the colour spaces. The normalized spaces are RGB, HSV and grayscales. Colour spaces that can efficiently separate the chromaticity from the luminance are generally preferred, since the use makes it possible to achieve some degree of robustness to illumination changes [1, 2]. Figure 2 shows segmentation that is currently usable on SPERA.

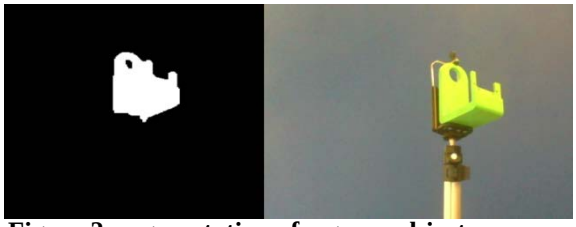


Figure 2: segmentation of a green object performed by SPERA, with the segmented image on the left and the original image on the right.

Skin colour has influence on recognition stability as well. The perceived colour of the skin varies between different races or even between individuals of the same race. This makes it impossible to make a separation on a specific colour. Additionally, variations in illumination and camera characteristics can influence colour separation greatly as well, which leads to the need of means to compensate this variability. In general colour segmentation can confuse objects in the background with the same colour as the human skin. A reduction of this problem might be the use of background subtraction. The background subtraction will however not work when there are too many similarities between the object and the background. [1, 2, 3]

The characteristic shape of both hands was utilized in research to detect the hands in multiple ways. Most of the information can be obtained using the extracted contours of objects within an image. If the detection is done correctly, the contour represents the shape of the hand and is therefore not directly dependant on skin colour, viewpoint and illumination. Contour detection often results in many edges that belong to the hand, but also edges from irrelevant background objects. Figure 3 shows what edges are created from an image. The amount of edges leads to sophisticated post-processing approaches to increase the reliability of the approach, since some of the edges are not necessary for the detection. Often colour is therefore combined with the shape for motion detection. [1, 2]

Figure 3: edge detection on a picture [34], with the original picture on the left and the edge detected picture on the right

Motion tracking has not been done often. In motion recognition there is assumed that only the hand in the image is moving, which demands a controlled setup. Using a combination of the motion in the camera with other cues has made it possible to better distinguish between the hands and other skin coloured objects and cope with lighting conditions. if the detection of the movement is fast enough to operate at the image acquisition rate, the detection can be used for tracking. Tracking hands however, has proven to be difficult since hands can move fast and change their appearance vastly in a few frames. Tracking can be defined as the frame-to-frame correspondence of the segmented regions. robust tracking is important as it provides the inter-frame linking of the hand appearances, giving rise to the trajectories of the features in time. These trajectories convey essential information of the gesture, which can be used for gesture recognition. [1, 2]

Vision based hand gesture recognition techniques can be divided into two subclasses, static gestures and dynamic gestures. To detect static gestures, a general classifier or template matcher can be used. Dynamic gestures require techniques that have a more temporal aspect to handle the dimension. Recognition can be achieved by using different machine learning approaches. The approaches that are used to classify or identify movement are computationally complex, especially when the system needs to recognise a great number of gestures. [1, 2]

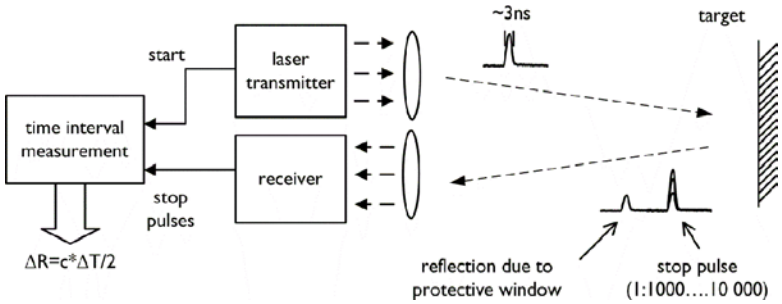


Figure 4: principle of time of flight measurements. A pulse of light is sent at time 0 by the transmitter, this pulse reflects on the object (target) and returns to the receiver.

Time-of-Flight (ToF) technology is one of the popular depth sensing techniques. With ToF the fundamental principle of light travel time is used to identify the movement. In Figure 4 there can be seen that a light pulse is sent out. This pulse reflects on an object and comes back in the receiver. Using the time between sending and receiving, the total distance can be calculated. The main advantage of ToF is the high refresh rate, while drawbacks are the resolution as this technique depends on the light power and reflection. Due to this resolution restriction this technique is currently especially popular on close distance hand and arm gesture recognition. [1]

Microsoft has the Kinect v2, which according to Microsoft uses the ToF principles. In the Kinect v2 there are two cameras, an RGB camera and an infrared camera. The infrared camera is used as ToF camera. The camera can perform the distance to object measurements for each pixel of its output data, resulting in a depth map. The RGB camera is used to create normal colour images. This colour image will then be translated to a colour map (separating the RGB data). The colour map and the depth map are then combined to create a colourized point cloud. This point cloud can be used to identify movement. [4] This technique, however, seems more like a distance calculation based on disturbances than using the ToF principles.

There have already been a few commercial products that made use of real time hand gesture recognition. Examples are the hand tracking devices of leap motion. These systems track hand movements and gestures and process the obtained information fast. The technology can be used for several applications within the field of augmented reality or virtual reality, without latencies that are perceptible for the users.

The previous research done with image-based motion recognition has mostly been about the recognition of hand movements. This is caused by the fact that a hand has some features that are better separable from other objects. Skin colour however, has been a cause of problems in object separation. When this type of recognition is used for arm movement tracking, there are some additional factors that make recognition

more difficult. Human arms are often covered in clothing, these clothes can vary widely in the colour, which generates more possibilities of similarities between background objects and the object that must be recognised. This makes separation more complicated than it is when only the skin colours must be considered.

For real time movement recognition there is a need of machine learning as well, since the movements must be tracked and interpreted, before the movement can be translated to angles to which the robot arm moves. This makes an image-based system less suitable for remote control of a robot arm for systems, since such systems need to be reliable and easy to use.

Non-image-based recognition approaches

Figure 5: IMU sensor glove by MI.MU, which contains one imu and several pressure sensors [33]

Image-based sensors were the main method used for gesture recognition for a long time. Since the recent developments in MEMS and other sensors, non-image-based recognition technologies have been increasingly popular. The most common approaches that are not using images are based on a glove or band that must be placed on the body of an operator. Currently google is developing another type of system that measures movement based on radio frequency (RF) waves to track the movements of a human.

Glove-based gesture recognition usually requires wire connections, accelerometers and gyroscopes. A glove with cables and other hardware for movement tracking often limits the ability to operate a system. This approach also requires complex calibration and setup procedures. An example of such a glove-based system can be found in Figure 5, which is a sensor glove created by the company MI.MU. This glove only uses one inertial measurement unit, for orientation estimation of the hand. The fingers in this glove are tracked using pressure sensors.

An alternative to these gloves is a wristband or other similar wearable device with sensors. These band-based sensor solutions often adopt wireless technology and electromyogram sensors to avoid cable connections. With band-based solutions user's hands are released, to increase the freedom of movement. [1]

MARG's are combinations of triaxial accelerometers, triaxial gyroscopes and triaxial magnetometers, contained in one package. The specific measurement techniques of each of these sensors can be found in Appendix A. [5]

The errors within the separate sensors of a MARG can be reduced. Some of the errors as bias and gain error can be compensated for with calibration of the sensors, since these errors are dependent on the temperature. Misalignment of the sensor axes can be overcome by calibration as well. Other errors of the sensors can be reduced by filtering the signal that is obtained from sensors. An example of these filters is the Kalman filter. Especially with MARG's, fusion of sensor data can be used to reduce sensor noise, drift, non-linearity and non-orthogonality.

Google's project Soli uses RF signals to track and recognise movement. With this technology a RF transmitter and receiver are used. The system selects a set of RF signals that can traverse through walls and reflect off

the human body. Based on the reflection time and angles, the movements can be recognised. Such systems could detect human motion from another room with a precision of 20 cm during the research of H. Liu et al., at this time Google states that their system is able to measure sub millimetre motion with high accuracy. This technique, however, is still in development and not yet ready for usage. [6]

At this moment there are numerous applications were hand held devices with MARG's are used to track motion, especially in the augmented or virtual reality field. Examples of these controllers are the controller from the google day dream [7] and the touch controllers of oculus which they deliver with their rift virtual reality headset [8].

There are numerous sensors available to be used for non-image-based movement detection. Most of these are IMU's, which offer 6 degrees of freedom (DoF). These commonly do not use a magnetometer and can therefore not provide full motion fusion. An example of this is the InvenSense MPU-6050, which only offers a triaxial accelerometer and gyroscope [9]. Within the group of 9 DoF MARG's there are a few options available to be used quickly at this moment. One of these offerings is the InvenSense MPU-9250 [10]. This module also includes a motion processer for data fusion, but this processor is limited by fusion of accelerometer and gyroscope data. Another option is the STMicroelectronics LSM9DS1. This is a 9DOF sensor which does not have any capabilities regarding data fusion. The third option is the Bosch BMF055, this sensor includes an arm cortex M0 and can deliver full sensor data fusion. This sensor is also called the BNO055 in integrated solutions [11]. Lastly there are several options from Xsens, however the price of Xsens sensors is with an average price of €200 per unit too high to be considered for this research, since €200,- is the budget for the complete system, which will need to involve multiple of these MARG's. [12]. There are currently no RF based movement sensors on the market at this moment.

Since sensors based on RF movement detection is still in development, it will not be possible to use this method yet. A system based on MARG's placed on the body of an operator is more promising at this time since there are several sensors available. The research done with this sensor type has mostly been based on single sensors, because in most applications such as controls in virtual reality, one sensor is sufficient. Since a robot arm requires higher accuracy, using multiple sensors spread across the arm of an operator might make movement tracking easier. In this case the angle of a joint from the human arm can be calculated using the data from two sensors, improving the capabilities in angle calculations. The problems created by the drift and accuracy of these sensors can be overcome using sensor data fusion algorithms and filters, such as the Kalman filter.

Joysticks for the control of a robot arm

Nowadays joysticks and controller are used in numerous applications within the gaming industry and in larger machines. In the gaming industry joysticks are often used in flight control. In these cases, a joystick is often limited to 3 DoF, the pitch, roll and yaw of a plane which are the front, side and twist movement for the joystick respectively. Game controllers are also used for these simulators, however for more experienced users the joysticks are preferred, since these give a better representation of the plane movement. The controllers mostly have 2 analogue sticks with each 2 DoF, front and side movement, resulting in a total of 4 DoF. The controllers are often used for drones, were one stick is used for the pitch and roll, the other stick is used for acceleration and yaw of the drone [13]. In larger machines joysticks are also used. In these cases, it is often used for control of a part of the machine, such as the arm of an excavator. Since these machines require more movement types than only the arm movement, often 2 joysticks are combined with foot pedals.

In automated industrial applications there is a lower interest in remote control, since the robots used here often must perform one set of movements repeatedly. There have however been tests with excavators, from which the control has moved from the cabin to a safe location for remote control of the excavator. In these cases, the excavator has been equipped with several sensors to give feedback to the operator that is on another location. [14] In robot control there have been several people that made a joystick or controller-based control system to control a simple robot. They also made instructions for others to make their own control system for such robots. [15, 16] What can be seen in most of these instructions is that when a controller is used for the movement of a robotic arm, the robot arm is limited to 3 or 4 DoF, since that can be managed easily.

A human arm has 7 DoF. The humanoid robot from Sogeti has this as well. Using a joystick to would lead to a lower freedom in movement of the robot arm, since the arm has more DoF than the controller can supply for. The limit can be overcome using the help of models and changing control of all separate joints to only the pitch, roll and jaw of the final position. The models will help in the other parts of the motion which cannot be done by the operator. This however makes control of the robot more difficult, since the operator only has control over the pitch, roll and jaw of the final position. The system thus requires more time for operators to learn to accurately control the robot with a joystick.

Improvement of sensor data.

In the non-image-based movement detection sub chapter there was explained gyroscopes and accelerometers suffer from drift by integration of the sensor data and noise by vibration of the sensor respectively. This often causes data to be inaccurate which makes it difficult to use these sensors for movement detection applications. Therefore, algorithms are required to improve the accuracy of the data. Most of these algorithms are based on sensor data fusion.

The Kalman filter, one of the options to use as a filter for sensor data fusion, can be used to help predicting values. Figure 6 shows the basic process. The Kalman filter is an iterative mathematical process that uses a set of equations and data inputs to make a better estimate of the values. The Kalman filter is based on gaussians, also known as normal distributions. The gaussian represents the predicted value with noise, error and uncertainty in the prediction, also known as the variance. Then the sensor data is used to update the state, after which the process restarts. The predicted value is based around the mean of the gaussian, with the width of the gaussian, denoting the uncertainty in the value. This basically tells whether a value is true or not. A larger width of the gaussian denotes that there is a larger uncertainty in the value [17].

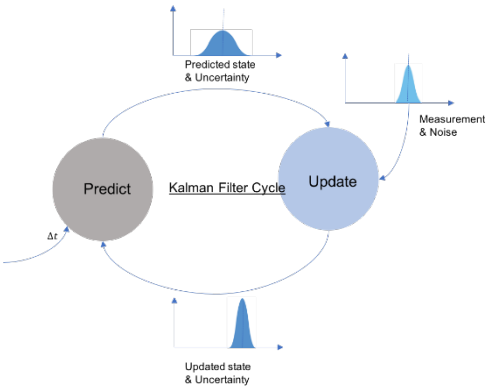


Figure 6: Kalman filter steps [48]. First a prediction is done based on previous data and the time step, afterwards this prediction is updated using the measurement data.

The process is based on two steps, prediction and updating. In the prediction, a new value is predicted based on the initial value. Afterwards the uncertainty, error and variance of the system can be predicted according to the process noises in the system. Then the value is updated taking the actual measurement into account. The difference between the predicted value and measured value is calculated, which is used for calculating the Kalman gain. Using the gain, the decision is made on whether the predicted value or the measurement is kept. Afterwards the new value and uncertainty, error and noise are calculated based on the decision from

the Kalman gain. These values become the predictions done by that iteration. The output is then fed back into the predict state which makes an iterative cycling process. [17]

The normal Kalman filter works on linear functions, however in the real world most systems involve non-linear functions, since several systems will look in one direction and measure in another direction. The extended Kalman filter is based on this problem. The Extended Kalman filter uses the first derivative of the Taylor series to approximate a non-linear function with a linear one. With the Taylor several derivatives are taken from a single point which in the case of the Extended Kalman filter is the mean of the gaussian. Using the first derivative of the Taylor series a tangent can be drawn around the mean to approximate the function linearly. [18]

With the Extended Kalman filter the prediction step is the same as that of the normal Kalman filter. In the update step there are some changes for the non-linear systems. For these systems the first derivative of the Taylor series is taken, which is also known as the Jacobian matrix. After that the values are converted to a linear space. Then the linear values are used in the same way as the normal Kalman filter does. [18]

The Unscented Kalman filter is another expansion on the Kalman filter. The Extended Kalman filter only uses one point, the mean, for the approximation of linear functions. Using multiple points will increase the accuracy of the filter, which is the goal of the Unscented Kalman filter. Transforming a complete distribution through a non-linear function is difficult. Using several individual points of the state distribution is easier. These points are the sigma points, which represent the complete distribution. More points will result in an approximation that is more accurate. These points are then weighted since the gaussian is an approximate. [19]

There are larger differences between the Unscented Kalman filter and the normal Kalman filter than between the Extended and normal Kalman filter. This is described in the Unscented transform. The Unscented transform is based on the following steps: compute a set of sigma points; assign weights to each sigma point; transform the sigma points through the non-linear function; compute the gaussian from the weighted transformed points; compute the mean and variance of the new gaussian. [19]

Due to all the calculation steps the Kalman filter is complex for hardware and difficult to understand. This causes difficulties in running the Kalman filter on small processors and getting the filter operational. There has been made use of the complementary filter to solve this issue. This filter is less complex than the Kalman filter, since there is a lower number of equations needed. The complementary filter performs both low-pass and high-pass filtering, which filters out the vibration noise from the accelerometer and the drift from the gyroscope respectively. [20, 21]

In other studies, the complementary filter has been adjusted to have better performance on MARG data. The study of Mahony et all changed the complementary filter to a state were the filter incorporates system dynamics using proportional and integral error estimates. This resulted in a lighter algorithm than the Kalman filter while it is more accurate and precise than the complementary filter. [22]

The filter based on an attitude heading reference system (AHRS) designed by Mahony performs the following steps to translate the sensor data from the MARG into the absolute orientation. First the inverse square root of all axis from the accelerometer are taken to normalize the magnitude of the measurements. This inverse square root is taken using the fast inverse square root algorithm. The same step is executed for the magnetometer as well. The filter calculates both the reference and estimated direction of gravity and the

magnetic field as the following step. This is then used to calculate the error between the reference and estimated directions, since the error is the sum of the cross product between the estimated and measured direction. The integral and proportional feedback is then applied on the gyroscope measurements. The rate of change of the quaternions is calculated and normalized using the corrected gyroscope measurements. With this result the orientation angles are computed from the quaternions. [22]

Madgwick et all designed another AHRS algorithm based on the gradient descent algorithm resulting in accuracy levels that match the Kalman filter, with < 0.8° static RMS error and <1.7° dynamic RMS error. The algorithm however has (like the Mahony algorithm) a low computational load, making it possible to reduce the hardware and power needed, thus increasing possibilities to use this filter for wearable devices. [23]

Madgwick's algorithm normalizes the magnitudes of the accelerometer and magnetometer in the same wat as Mahony's algorithm. The reference direction of earth's magnetic field is calculated based on the magnetometer measurements, which is then used in the gradient descent algorithm to perform the corrective steps to the algorithm based on the sensor data. Then the rate of change in quaternions is calculated based on the gyroscope measurements. The feedback obtained from the gradient descent is applied to the rate of change in quaternions. Afterwards the rate of change in quaternions is integrated to yield the quaternions. The orientation angles are then computed from the quaternions. [23] Other research done on the sensor data accuracy and filter accuracy provided that with well calibrated sensors the AHRS algorithm from Madgwick can obtain a 4° accuracy. [24]

The gradient decent used in Madgwick's proposes algorithm is an optimization method used to find the local or global minima of a function. This algorithm is used to iteratively update the weights of a function to estimate the lowest error function. In Madgwick's research the Stochastic gradient decent algorithm is used, since this only relies on small batched of samples for each iteration. Larger sample sizes would result in increased computation times which in its place increase the computational load. [23, 25]

There are multiple implementations of the algorithms proposed by Madgwick and Mahony their research. With the research of Madgwick a proof of the algorithm has also been provided. The implementations of these algorithms are based around open source licences and can thus be used for other research.

ROS, an operating system for robots

The robot from Sogeti users the Robot Operating System (ROS) framework. ROS is a modular and distributed framework that is made to control robots, this enables users of ROS to make their own choice between the available parts and parts that they prefer to build on their own. The distributed nature of ROS adds large community support of contributed packages on top of the core system. This allows for wide usage of ROS. [26]

ROS uses different 'nodes', these nodes can be made to have their own purpose and are easily added to the system. The system is based on a master node that handles the registration of smaller nodes. The master generates a lookup table based on the registered nodes. This lookup table makes it possible for nodes to communicate without the need of an IP address. All nodes in a system can send direct messages to each other using a publish and subscribe model. Nodes can publish their messages on a topic, other nodes can subscribe to this topic to obtain the information from the publishing node. [27]

In Figure 7 the nodes have been displayed by ovals and the topics by rectangles. This figure represents the basic communication that is used to operate vision-based object tracking on SPERA. Ros has additional safety

features installed, if one of the nodes stops functioning, the other nodes will still work, however a specific task might stop functioning. In the case displayed in Figure 7, if the image processor stops functioning, the motor controller of the head will still work. The head would only not be able to follow the object, since it is not recognized.

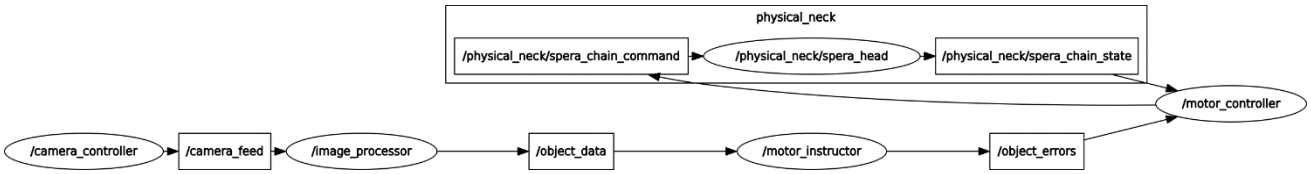


Figure 7: basic overview of ROS interactions, within the current object recognition system of SPERA. The oval shapes are nodes, the rectangles the topics that these nodes are either subscribed to or publish onto.

Angle conversion

After filtering the signal with one of the data fusion algorithms, the resulting data will be in the roll, pitch, yaw angle format. This is a specific order of Euler angles. This order is the ZYX angle order were the rotation along the Z axis results in the Yaw information, rotation along the Y axis in the Pitch and rotation along the X axis in the Roll of the sensor module. This data had to be translated into the angles that the robot arm uses. There has been little research done into the conversion of MARG angles to the angles of a robot arm, therefore an approach to convert these angles has been proposed during this research.

Hypothesis.

At this point none of the options for remote control of a robotic arm are perfect, however, when taking the clients requirements and the possibilities of each control type into consideration, the most likely solution of this problem will be based on MARG's. The reason for this is that Sogeti would like to see a solution that can be implemented easily on different users, with a small learning time for controlling the system. Furthermore, they would like to have a system that can operate in all lighting conditions and is not dependent on skin colour, since that would limit the number of people able to use the system. Therefore, joystick-based control and movement tracking based on image recognition are less suitable, since these options are either highly dependent on lighting or have higher learning times. RF movement detection systems are currently not available, causing them to not be useable for this research. A decision matrix displaying the more detailed choices can be found in Table 2.

Table 2: decision matrix for the recognition method

	Image based	MARG's	Joystick
Ability to reconstruct arm motion	++	++	[]
Ability to handle environmental influences			
Dependant on lighting/colour conditions	--	++	++
Ability to handle multiple persons near measurement device	--	++	++
Operator dependant factors			
Learning time to operate system	++	+	--
Ease to switch between operators	++	--	++

++ best; + better than average; [] average; - worse than average; -- worst;

Using the requirements from the client, user stories and the most likely approach from the literature described in the previous chapters, the set of functional system requirements shown in Table 3 and the set of non-functional requirements shown in Table 4 have been formed. The user stories and storyboards which were used to obtain these requirements can be found in Appendix B.

Table 3: functional requirements for the sensor system based on user requirements, customer requirements and literature, for the final system

Number	Requirement	Priority
SF1	The system shall enable the robot to mimic human arm motion	Must
SF1.1	The system shall produce the parameters needed for movement of the robot arm and gripper	Must
SF1.1.1	The system shall measure the angles of the human joints, which it converts to the angles for the robot arm.	Must
SF1.1.1.1	The system uses 4 accelerometers for movement of the robot arm	Must
SF1.1.2	The gripper from the robot must be open or closed	Must
SF1.1.2.1	The system uses a button for movement of the robot's gripper	Must
SF1.2	The system shall give accurate movement in all lighting conditions	Must
SF1.2.1	The system shall not use cameras to mimic a human arm.	Must Not
SF2	The system shall be battery powered.	Must
SF2.1	The battery shall be protected using a battery management system.	Must
SF3	The system shall communicate wirelessly with the robot	Must
SF3.1	The robot and the system shall be separate systems that communicate with each other	Must
SF3.1.1	The system shall use the wireless protocol that is already in use for the robot	Must
SF4	The system must be able to be disabled temporarily.	Should
SF4.1	The system shall have a switch for temporal disablement	Should
SF5	Placement of the sensors shall not limit the movement of the operator	Should
SF6	The system shall be able to put on and taken off the operator easily	Should
SF6.1	The system will contain connectors at each sensor for easy connection and removal of wires	Should

Table 4: non-functional requirements for the sensor system based on user requirements, customer requirements and literature

Number	Requirement	Priority
NF1	The system must comply with the low voltage directive	Must
NF2	The latency must be lower than 300ms	Could
NF3	A new user must be able to learn how to operate the system within 15 minutes	Could

Furthermore, with the availability of implementation and proof of Madgwick's algorithm, the performance that is comparable with the Kalman filter and the efficiency for the algorithm on hardware, means that Madgwicks algorithm will be the best option for a proof of concept and therefore will be used during this research. Since the priority is set on creating a proof of concept for the system, the system latency will not be tested, literature does describe that a latency between 150 and 300 ms is preferred. This literature can be found in Appendix C.

Conceptual model

At this moment there are several options for remote control of a robotic arm. These options range from movement tracking with image recognition, wearable sensor systems using MARG's, RF movement detection and joystick controllers. Each of these systems have their drawbacks. Although image-based movement recognition has become better in recognising movement in different lighting conditions, the colour of skin and clothes complicates using these systems by different operators [1]. MARG's have problems with their accuracy caused by drift, which decreases performance [5]. RF movement detection offers a lot of potential for movement recognition systems, since they can separate humans from the background using specific frequencies that pass through most objects except humans [6]. Currently however, such systems are still in development and not ready to be used. Joystick based systems have limited control over the robot since they offer a lower DoF. This can be solved by combining the joysticks with models for the arm movement, which leads to an increased time to accurately learn to control the robot arm.

Sogeti would like to see a solution that can be implemented easily on different users, with a small learning time for controlling the system. Furthermore, they would like to have a system that can operate in all lighting conditions and is not dependant on skin colour, since that would limit the number of people able to use the system. Therefore, joystick-based control and movement tracking based on image recognition are less suitable. Since RF movement detection systems are not available currently, this will not be useable. Currently the most likely solution to this problem is the usage of multiple MARG's spread over a human body, since these sensors are not limited by the images created and normal motion can be used for the robotic arm movement. Figure 8 shows the possible placement of these MARG's. Since MARG's face accuracy problems, the data of these sensors must be fused using a sensor data fusion algorithm.

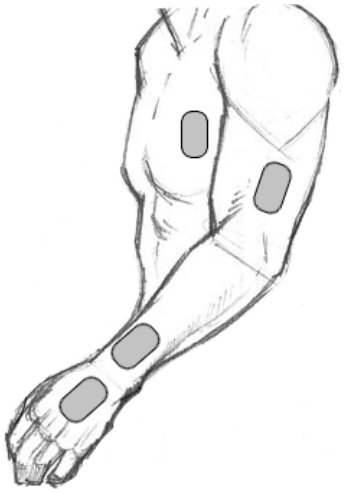


Figure 8: proposed IMU sensor placement. One sensor on the torso, one on the upper arm, one on the lower arm and one on the hand.

For sensor data fusion there are multiple options, one of the Kalman filter types and or the complementary filter. There are 3 commonly used types of the Kalman filter, the original one, the Extended Kalman filter and the Unscented Kalman filter. Gyroscopes in a MARG perform measurements in another direction than the movement happens, this results in sensors that are non-linear. The original Kalman filter can only be used with linear sensors [17], therefore the original Kalman filter is not suitable to increase sensor accuracy. The Extended and Unscented Kalman filter are both able to compensate for the non-linear behaviour of these sensor systems. The Extended Kalman filter does this using the mean of the gaussian and then converts a non-linear function to a linear approximate using the first derivative of the Taylor series [18]. Afterwards the Extended Kalman filter behaves like the original Kalman filter. The Extended Kalman filter only uses one point (the mean) to generate the gaussian. The Unscented Kalman filter is based on multiple weighted points, called the weighted sigma points. The usage of multiple points increases the accuracy of the algorithm [19]. The algorithm is however more complex for the Unscented Kalman filter when compared to the Extended Kalman filter, since there are more variations from the original Kalman filter. The Kalman filter is complex in usage and understanding causing it to be difficult to be used on small controllers. To solve this issue Mahony and Madgwick proposed algorithms that are based on the complementary filter. These filters are both widely available under open source licenses and light for hardware. Since the research of Madgwick proved that the accuracy is close to that of the Kalman filter, this will be the used option.

The robot from Sogeti operates on ROS. This is an operating system build around the concept of modularity and a distributed nature. The configuration of ROS is based on a node system, it has a master node which handles the registrations of the nodes. Furthermore, each node is not aware of the existence of other nodes. A node can be subscribed or publish on a certain topic. This way it can either send or receive messages containing the information that is needed [27]. The new system can be connected to the existing system and

publish on the topic to which the node that calculates the movement of the robot arm is subscribed, after the new system has been validated and works as expected.

Latency is an important factor in remote control. An ideal system would have zero latency, real world applications however, do have latency. The latency in a system determines a large part of the perceptiveness for users [28]. This research will not focus on the latency of the combination between the new system and the robot, however since there are limits for users to perceive a system as useable, the latency will be considered. Sogeti uses guidelines for the maximum latency that a system can have for a good user experience. The combination of the new system and the robot will be tested to see whether it can perform below the latency limit. This will test the viability of the new system based on the user experience.

Research design

The design and validation of the system has been done according to the V-model. Figure 9 shows the steps that have been taken for both the design and validation of the system. With the V-model each design step was validated with testing steps. This model is also called the verification and validation model [29].

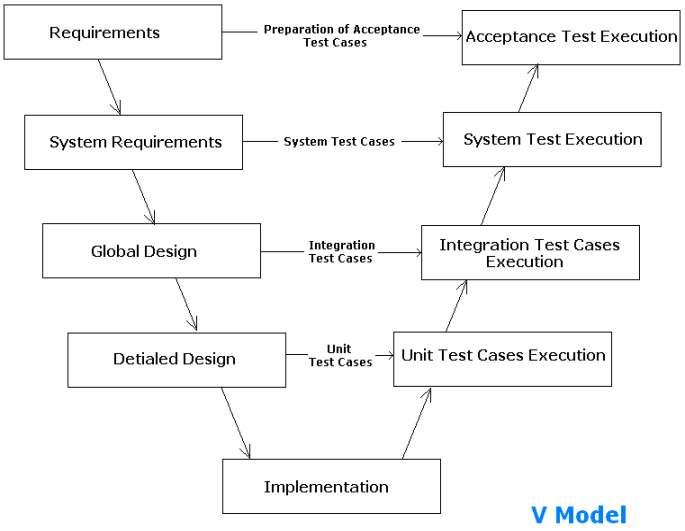


Figure 9: V- model for development of systems [29]. This model includes the steps that have been performed to create the system proposed in this research.

Requirements

The system requirements have been made based on a combination of the story board, user stories and user requirements that can be found in appendix A. The complete list of requirements for the final product can also be found in the hypothesis section of the theoretical analysis. This list is also prioritised based on a complete system. For this research however, other priorities have been set. The list with the requirements on which the research will focus can be found in Table 5.

Table 5: requirements for the proposed remote-control system, prioritized based on the needs for the research.

Number	Requirement	Priority
SF1	The system shall enable the robot to mimic human arm motion	Must
SF1.1	The system shall produce the parameters needed for movement of the robot arm and gripper	Must
SF1.1.1	The system shall measure the angles of the human joints, which it converts to the angles for the robot arm.	Must
SF1.1.1.1	The system uses 4 accelerometers for movement of the robot arm	Must
SF1.1.2	The gripper from the robot must be open or closed	Must
SF1.1.2.1	The system uses a button for movement of the robot's gripper	Must
SF1.2	The system shall give accurate movement in all lighting conditions	Must
SF1.2.1	The system shall not use cameras to mimic a human arm.	Must Not
SF2	The system shall be battery powered.	Could
SF2.1	The battery shall be protected using a battery management system.	Should
SF3	The system shall communicate wirelessly with the robot	Could
SF3.1	The robot and the system shall be separate systems that communicate with each other	Should
SF3.1.1	The system shall use the wireless protocol that is already in use for the robot	Could
SF4	The system must be able to be disabled temporarily.	Could
SF4.1	The system shall have a switch for temporal disablement	Could
SF5	Placement of the sensors shall not limit the movement of the operator	Could
SF6	The system shall be able to put on and taken off the operator easily	Could
SF6.1	The system will contain connectors at each sensor for easy connection and removal of wires	Could

During this research focus has been on creating a proof of concept for a measurement system to translate human arm movement into angles needed to move the robot arm. Therefore, it was most important to create a working sensor system and put a lower priority on things like wireless communication.

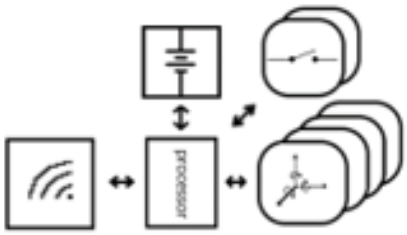


Figure 10: global system architecture for the hardware of the system. The sensors should be connected to a controller that processes the data and work wirelessly.

Global design

Hardware

For the hardware there was chosen to base the system on 4 MARG sensor modules connected to a controller. Furthermore, there should be one button to control the gripper and one switch to turn the remote control on or off. The system should communicate wirelessly with the robot and be battery powered. Figure 10 shows the global hardware design for the system.

Software

The system has been designed around the ROS node concept, so less time was needed for integrating the separate parts into the complete system. The nodes were designed to all have their own input, which was either sensor data or information obtained from one of the ROS based topics. The data published on the topic was specialized for the function of the node. The parts displayed under the part of the Robot (SPERA arm and SPERA gripper) in Figure 11 were the existing systems which have not been altered, thus the output of the last nodes before the robot nodes, were pre-defined. The output of the other parts was based on the logical output and input of these nodes. Figure 11 shows an image of the global system architecture. The ovals that are displayed in this figure are the nodes which have been made for the system, the squares were designed to be sensor input and the text above the arrows show the type of data published on the topic. Further explanation of every node has been done in the software part of the detailed design.

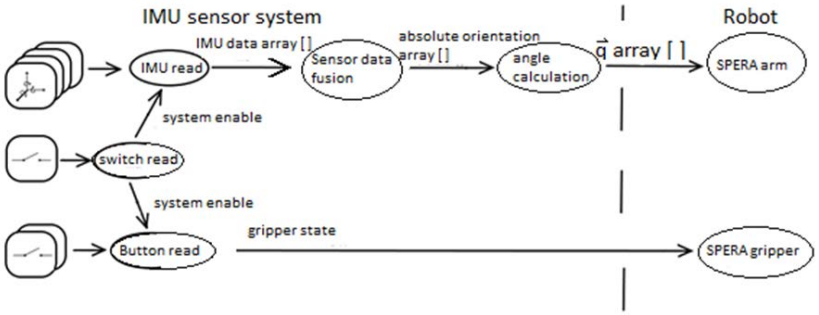


Figure 11: System architecture of the sensor system. The parts under IMU sensor system are the parts that have been built during this research. The parts under Robot were the parts that were made before this research.

Detailed design

Hardware

ROS only runs on Linux based systems. Since the SPERA is based on ROS, for an easy integration it would be best for the new remote arm control system to run on the same software as well. Since the Raspberry Pi contains both general purpose input/output (GPIO) pins to read sensor data and can run Linux based operating systems a Raspberry Pi was chosen as the controller for the new system. Although in the requirements it was noted that the addition of wireless communication and having the system battery powered was on a lower priority, the addition of these parts was not time consuming and has therefore been added to the hardware as well. The used controller, the Raspberry Pi, has a wireless network controller built in since the Raspberry Pi 3. Therefore, this controller has been used during the project. A power bank was used as battery, since power banks are supposed to be used with portable devices and have safety

features such as over discharge protection built-in. the specific model used was the Anker power pack 5000, due the small size in combination with the delivered capacity and the output current.

MARG's from different manufacturers have been compared on characteristics, interfacing and pricing. 3 sensors modules from different manufacturers were compared: The InvenSense MPU-9250, the STMicroelectronics LSM9DS1 and the Bosch sensortech BNO055. The results have been described and the final sensor choice has been supplied. Additional options were available, but these were either not available within the budget of €200, - for the system or the utilised communication protocol did not allow the use of more than one sensor module of the same type.

All sensors that have been compared work on the I^2C interface. However, they are all limited to one or two hardware addresses, which is not enough for the required amount of 4 sensors. This problem can be solved with an I^2C expander, but this increases the complexity of the system, which is unwanted. The alternative interface on the Bosch BNO055 is UART. There can only be one device on a UART interface, so this interface is not suitable for the application either, since this would need 4 UART interfaces which are not standard available on the Raspberry Pi. The MPU 9250 from InvenSense and the LSM9DS1 from STMicroelectronics both use the SPI protocol as secondary output. This protocol can be used with a larger number of sensors since each sensor has its own chip select pin. The Raspberry Pi officially only offers 2 hardware chip select pins. In the SPI protocol the chip select only changes state to select a chip. Since this is a simple digital pin operation, this function can also be mimicked by the standard GPIO of the Raspberry Pi. This way four sensors could be used with this protocol.

The sensitivity of the InvenSense MPU9250 and STM LSM9DS1 is the same, since they both use a 16 bits ADC for the accelerometer, while the Bosch BNO055 has a 14 bits ADC on the accelerometer. The sensitivity of the gyroscopes and Magnetometer are the same for the 3 types. There are larger differences in the accuracy of the sensors. In most cases the Bosch BNO has the highest accuracy the InvenSense MPU has an accuracy close to the Bosch BNOs, and in some aspects even surpasses the Bosch BNO. The STM LSM9DS1 has little data available in its data sheet about the accuracy and drift. The data available about this sensor shows this sensor will be less accurate than the other two options. The price is one of the more important aspects and shows a larger difference between the 3. The Bosch BNO starts at €25, - for large modules, but can mostly be found for €36,95. The STM LMS9DS1 can be obtained for €17,95 and the InvenSense MPU can be obtained for €8,50. There are multiple libraries available for each of the sensor modules, but the support for the Bosch BNO with a Raspberry Pi is little. Libraries containing register maps are more common on the Raspberry Pi for the other 2 sensor modules. Furthermore, all the sensor modules use the same logic level as the Raspberry Pi, which is 3v3. The combination of these characteristics made the InvenSense MPU9250 the best choice of the three sensors. In appendix B, further details on the comparison and detailed specifications can be found.

Software

The detailed design of the software has been made according to the system architecture displayed in Figure 11. The nodes named in this subchapter can be found in this figure.

System enable switch

This node had been incorporated into the design but would only be made when the higher priority parts were working. The system enable switch has been used to note whether the robot arm needed to move or whether the robot arm was supposed to stop moving. When the state of this switch was 0, the robot arm received a command to move to the rest position. In the other case, the robot arm moved according to the

movement of the human arm. The state of the switch was published on the switch enable topic, such that this information was available for the other nodes.

Gripper state button

The gripper state node measured the state of a button. After a change in the state of the button, this change was debounced to prevent multiple changes occurring within a short time. The debouncing was done based on time. After a press of the button, the state of the button changed from high to low. When this change occurred, the system measured whether the change in state was kept for at least 10 cycles (which results in a few ns on a Raspberry Pi. After the debouncing concluded that the button was pressed, the time of the press was recorded. A press that took under 2 seconds resulted in an open gripper and a press longer than 2 seconds resulted in a closed gripper. After the button was released, the change was debounced again to prevent the button bouncing at release as well. The new state was then published onto the “gripper_state” topic, so that the gripper of the robot moved based on presses of the button.

MARG data read node

The MARG read node used the sensors and the switch state as input. The SPI protocol was used to communicate with the sensors. Each sensor had its own chip select (Cs) pin. Before a sensor was read, the chip select of that sensor was set low. This activated the sensor such that the data could be obtained. Obtaining the data was handled by a combination of the WiringPi library and the MPU9250 SPI library. The MPU9250 SPI library contained the addresses of the sensors from which the sensor data was read and the calculations to translate the raw sensor data to units that were wanted. The WiringPi library handled the pin control for the Raspberry Pi. After the data for the 4 sensors was obtained, the data was combined into one array and published on the “multi_MARG_data” topic. The switch state was forwarded to the filter node, since the system needs to give the arm the right commands when the system is disabled, while the system kept updating the filter, to accurately track the human arm.

Sensor data filter node

The sensor data filter node was based on an application of the filter presented in the paper of Madgwick et all [23]. The node used the MARG sensor data array as input. There was a separate instance of the filter for every sensor, which updated every time new sensor data arrived. The filter works using a gradient decent algorithm that combines the gyroscope data with the gravitational force of the accelerometer and the magnetic field measured by the magnetometer. After the filter calculated the absolute orientation for the 4 sensors, the orientations were published as an array.

Angle conversion node

The angle calculation node used the absolute orientation arrays as input. This data was then split into the separate sensors, after which the data was transformed into the rotation matrices relative to the earth, since the measurements of a MARG are relative to the earth. The rotation matrices relative to the earth were then transformed to the rotation matrices relative to the previous body. The angles of the joints were then calculated based on the rotation matrices relative to the previous body.

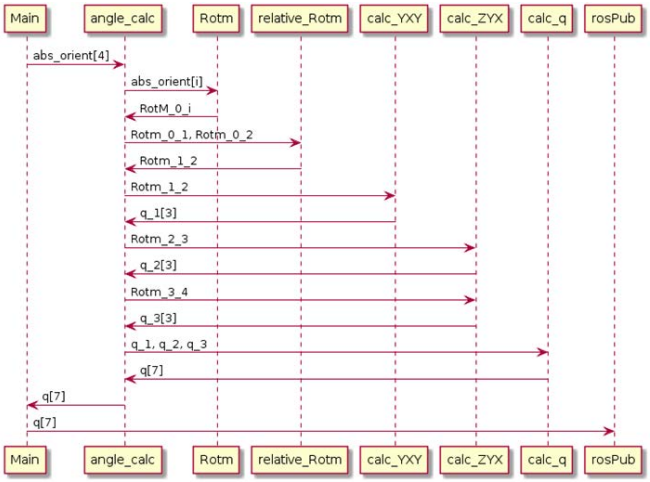


Figure 12: u.ml diagram for the angle calculation node, describing the steps that the software will take

A u.ml graph has been made for each node describing the steps that the software had to perform. Figure 12 shows the u.ml graph for the angle conversion. The u.ml graphs for the other nodes can be found in Appendix C

Implementation

Hardware

In the design phase of the project, the choice was made to use the MPU9250 from InvenSense for the sensor modules. The specific modules selected were the generic imported modules from China, which in the Netherlands can be obtained at a price of €8,50 per module. In total 4 of these modules were ordered and small connection PCB's were etched for them. However, due to a problem at the supplier different sensor modules were used. The modules that were used instead were still based on the MPU9250, but on a break-out board from Sparkfun. This was the best alternative since the price was like the LSM9DS1 at €18,95 and this required the least changes to software.

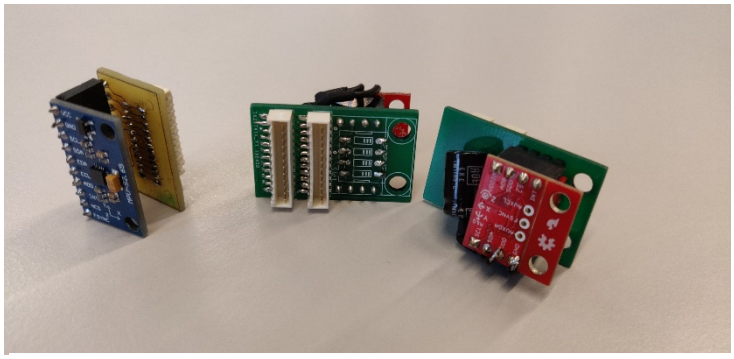


Figure 13: difference between the old modules (blue PCB) and the new modules (red PCB). The green PCB's are the boards that have been manufactured by JCLPCB, to make a modular design

The PCB's used to connect the separate cables needed large changes however, since the pin layout of the new module was different. This difference can also be seen in Figure 13. After the design of the new boards was made, the boards were fabricated by JLCPCB, since this company delivered professional quality PCB's which arrived within a week at a price of €7, -. Before ordering, the choice was made to add another change to the small PCB's, in the first design each PCB had a separate chip select connected, causing each module to have a specific board. The new modules were designed such that all PCB's for the sensor modules could be the same. To do this, while still making it

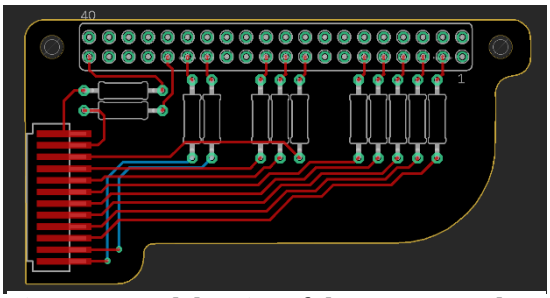


Figure 14: Cad drawing of the PCB created to go on top of the Raspberry Pi. In the design, all resistors used where 10 ohms.

possible to have a separate chip select per module, a dip switch was added to the design. The connection to the chip selects of each module could then be done by sliding the specific switch. When the new boards arrived, the components were soldered on top and the cables were made. The cables used to connect the 4 sensor boards and the Raspberry Pi have been manually made, with lengths dependent on the placement of the modules. The sizes of the cables where roughly 20, 30, 40, 60 cm. After which it was possible to test the new sensor modules.

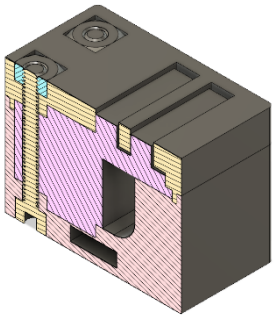


Figure 15: Assembly of the modeled sensor enclosures including a model of the sensor module and the bolts used to close the casing.

Next to small PCB's for each of the sensor modules, a PCB was made to go on top of the Raspberry Pi as well. This board was used to connect the GPIO pins on the Raspberry Pi with the same connector as was used on the smaller boards. Furthermore, 10 ohms resistors were used on this boards to provide some pin protection for the Raspberry Pi against electrostatic discharge. A CAD drawing of this design has been provided in Figure 14.

The resistance of 10 ohm resulted in problems on longer cable distance for the MARG modules. Although sensor data for the accelerometer and gyroscope was read correctly, the magnetometer was only reading data for the first seconds. After these few seconds, the magnetometer stopped generating new data. Changing the resistors on the data and clock lines of the SPI protocol (MOSI, MISO, SCLK) to 80 ohms, to better fit the total line impedance, solved this issue.

After there was noted that the sensors were giving the expected results, the choice was made to design cases for the sensors, for a better way to mount the sensor modules on the arm. The cases were printed with a 3D printer, which was an Ultimaker 2 extended. The design included the shape of the sensor modules, such that the modules fit tightly inside the casing. This way the sensor modules were not able to move around in the casing. Additionally, holes were added to secure the sensor inside the enclosure with bolts and another cavity was added for a piece of Velcro strap to mount the module on the human arm. Figure 15 shows the casing in the assembly, connecting all pieces of the casing in software. The printed enclosure from multiple angles including the inside can be found in Figure 16.

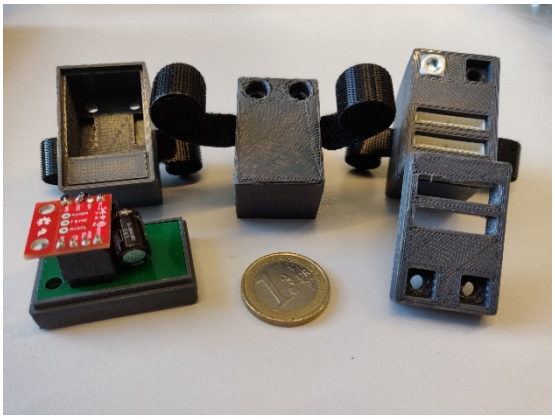


Figure 16: The printed sensor module casings, with a €1, - coin for size comparison. The sensors fit inside the enclosures without being able to move around.

System design

Pictures of the completed system can be found in Figure 18 and Figure 17. In Figure 18 the system can be seen with all sensor modules next to each other, while connected to each other and the Raspberry Pi. In Figure 17 the system can be seen while mounted on a human arm. The placement of sensor one on the torso and the Raspberry Pi in Figure 17 was not final. The placement used was causing movement in the sensor, thus generating inaccuracies in the sensor data.

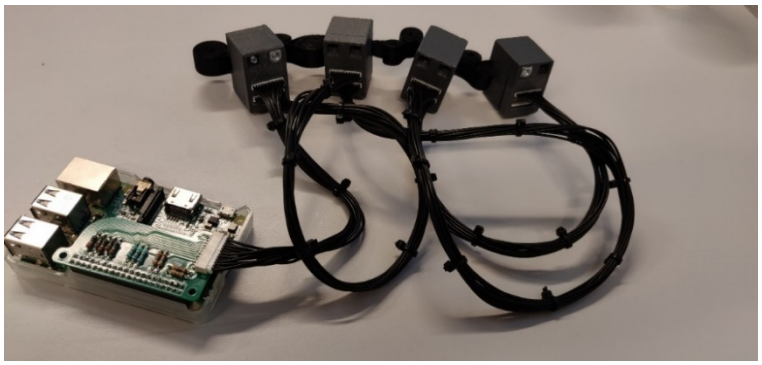


Figure 18: the completed sensor system with the cables connected to the modules and the Raspberry Pi.

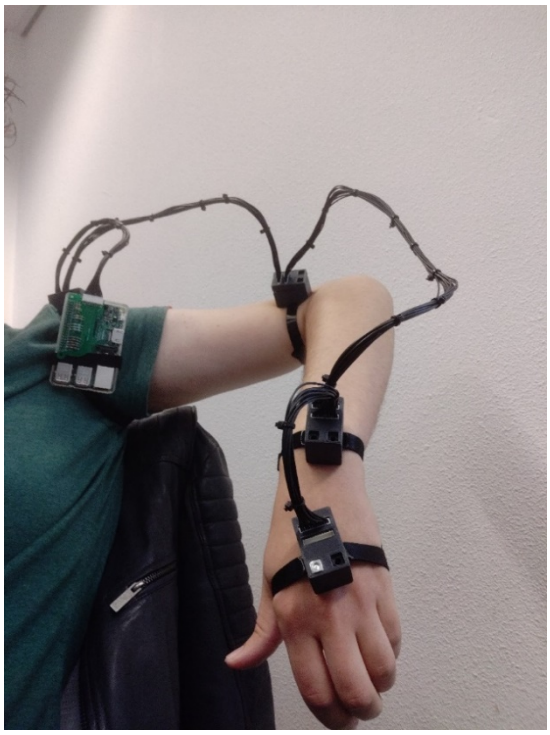


Figure 17: the sensor system mounted on a human arm. The cables are chained and can be removed.

Sensor calibration

The sensors were connected to the Raspberry Pi and the software of the sensor read node (described in the software part of this chapter) was started. The sensors gave promising results but had unwanted biases (offsets). Especially the magnetometer needed calibration for this. Calibration of the accelerometer and gyroscope was done by getting a data set from 15 seconds of data at a sample rate of 100 Hz, thus 1500 samples, when the sensor modules were in a static position. For the bias calculation of the accelerometer and gyroscope the mean of this data set was taken.

The bias of the magnetometer was calculated in a different way. The data set was obtained while moving the module in an 8 figure in different directions. Using this method, a 3d image of measurement points could be taken for the magnetic field. The bias was then calculated based on the mean of the maximum and the minimum of the sensor readings. For the magnetometer the scale factor was calculated as well, this was done by first calculating the range from each sensor axis. The ranges of the tree axes were then combined to calculate the mean of the tree ranges. To obtain the scale factor of each axis, the mean of the tree axes was then divided over the range of that axis.

The calibration that has been done for the magnetic field is dependent on the surrounding magnetic field. In the case of outside using the surrounding field will not generate large changes. In buildings like offices however, there can be many changes to the field. Since the proposed measurement system was supposed to be used inside buildings, calibration needed to be done more often. To achieve this possibility, two small programs have been made. Both programs collect a dataset of 1500 samples which are used for the calculations. One of these datasets could be used to generate the bias of the accelerometer and gyroscope, while these are placed in a static position. The other program has been used to collect data from the magnetometer while the sensor is moved around in an 8 figure as was done for the initial calibration. The

calculated biases and scale factors (in the case of the magnetometer) were then saved into a text file by the program. Afterwards the sensor data read node could read the files to set the biases and scale factors at startup.

The choice was made for this calibration approach, since the data generated for calibration of the magnetometer while the system was mounted on the arm would not generate an accurate bias and scale factor. The reason for this is the usage of the maximum and minimum of the measurements along all rotations to calibrate the sensors.

Software

Before usage a fresh installation of Ubuntu mate 16.04 was installed on an SD card for the Raspberry Pi. The reason for this specific OS was that Ubuntu mate works well with ROS on a Raspberry Pi and that the version of ROS used with SPERA, ROS Kinetic, only works with ubuntu release 16.04. After the installation of Ubuntu mate was finished, the full version of ROS Kinetic was installed on the Raspberry Pi, using installation instructions provided by the ROS page [30]. After the installation of ROS was finished, WiringPi was installed according to the instructions on their site [31]. After installing these packages, the Raspberry Pi was ready for use. The gripper state node was made according to the design without changes needed.

Sensor read node

At first the plan was to use a library for the MPU sensor module that was made to work with the Raspberry Pi, there were however multiple problems with this library, causing it to only read the data from the accelerometer and a part of the gyroscope data. Because of these problems there was chosen to edit a library that has been working with an Arduino microcontroller to work with the Raspberry Pi. The library that was used was created by Bolder Flight Systems. After changing the library to work with the WiringPi pin library of the Raspberry Pi, problems occurred regarding the usage of normal GPIO as chip select pins. After some testing and using the data sheet from the MPU9250, there was noticed that there was a time limit after which there needed to be written to a sensor after a change on the chip select. This time limit was 25 ns. Using the normal pin commands with WiringPi, appeared to be too slow. This is caused by the fact that the normal pin commands in WiringPi use the same layers as the Arduino pin commands, causing more time to set GPIO pins. WiringPi also offers functions to alter GPIO pin states on hardware level. Pin changes based on the hardware level are faster, making it possible to use these GPIO pins for the chip select for the SPI protocol. However, use of these functions is protected and needed a special command before a pin can be used with this method. The command that needed to be given was “gpio export 6 out”, which enabled hardware pin 6 to be set as output. After this change it was possible to address the sensors on the normal GPIO as chip select, thus the 4 sensors could be used on one Raspberry Pi.

The data from the new library did have the information in the wrong units. The acceleration was in m/s^2 instead of g, the rotation was in radians per second instead of degrees per second and the magnetic field was represented in micro Tesla instead of Gauss. The calculations to the units were changed inside the library, such that no calculations had to be performed on the sensor data after it was obtained from the library.

SPERA runs at an “heartbeat” of 100 Hz. This means that the software updates at 100Hz. Therefore, there has been chosen to let this node initialize all sensors once and read the calibration data from the text file for the given sensor. After the initialization, the sensor data for each of the four sensors was obtained at 100Hz. The data of the 4 sensors was then combined into one array such that the data could all be published at once.

Sensor data filter node

After the sensor data read node was created and was giving the expected data as output, the data filter node was made. This node was based on an open-sourced implementation of Madgwicks AHRS algorithm. With this implementation the filter used acceleration in g, rotation in degrees per second and magnetic flux in Gauss. Since the output of the data read node was already changed to these units, there was no need to perform changes to the sensor data. A separate instance of the filter was made for each sensor such that the sensor When a new message was arrived, the message was split into the separate sensors and passed through the instance of the filter for that sensor. The resulting orientation in radians was then added to a new array, such that the orientation for the 4 sensors could be published together again.

Angle conversion node

Most sensor data fusion algorithms use roll, pitch and yaw as output data format, since this is the most common representation for orientation data. Orientation using the roll, pitch and yaw is given in Euler angles with the following order of rotations: the first rotation is along the Z axis, which generates the yaw angle. The second rotation is along the Y axis, which is the pitch and the last rotation, along the X axis, which generates the roll angle of the sensor. The order of rotations for orientation of the sensor data is therefore given by the Euler angle order ZYX. The robot however uses a different set of Euler angle order combinations for the arm.

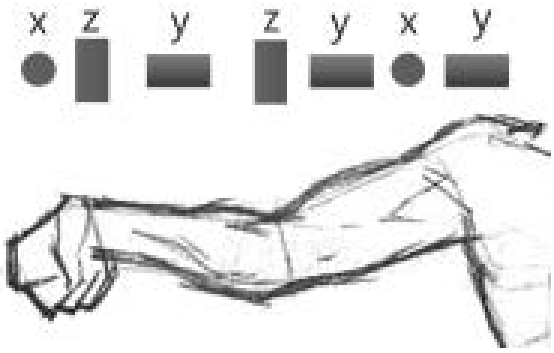


Figure 19: schematic representation of the position in which both the angles for the robot arm and the human arm are 0, including a representation of each rotational joint. The resulting angle order is YXYZYZX.

The robot arm is made to have an angle of 0 on each joint when it is positioned in a straight line, as can be seen in Figure 19. The first rotation for the robot arm, starting at the torso, is a rotation over the Y axis. The second rotation, the raise movement of the arm was over the X axis. The third rotation, the twist of the upper arm was again over the Y axis. The bend of the elbow, the fourth joint was a rotation over the Z axis. The twist of the lower arm was another rotation along the Y axis. The side movement of the wrist was another rotation over the Z axis. Lastly, the raise movement of the wrist was a rotation along the X axis. This results in an order of angles which is YXYZYZX, which is the same angle combination as the human arm, if the assumption is made that the angles for the human arm are also 0 in a straight line.

Each of the 4 MARG's were responsible for generating data of specific joints of the human arm the first sensor placed on the torso would generate the data of the rotation of the torso relative to the earth. The second sensor measured the rotation of the upper arm relative to the earth, placed after the third joint, thus delivering the YXY angles. The third sensor measured the lower arm relative to the earth, after the fifth joint, for the ZY angles. The last sensor measured the rotation of the hand relative to the earth. This resulted in 4 sets of orientation data, were all measurements were relative to the earth. Since the angles of each joint set based on the previous set were wanted, calculations had to be done to convert the angles relative to the earth to the angles relative to the previous sensor.

Since all the joints in the robot were rotational joints, only calculations based on rotation matrices were needed. The first step was to generate the rotation matrices based on the sensor data. Since the sensors measure relative to the earth, the generated data was the rotation of a sensor module compared to the earth. Each of the four rotation matrices generated were based on the sensor's measurement relative to the

earth, so the sensor on the torso was measuring R_1^0, the sensor on the upper arm was measuring the R_2^0. And similar for the other two with R_3^0 on the lower arm and R_4^0 on the hand. For the angles of the joints however, the difference between the previous and the sensor of interest was wanted, so the rotation matrix $R^1{}_2$ which is the difference between R_1^0 and R_2^0. The rotation matrix of R_2^0 can be obtained by the following formula: $R_2^0 = R_1^0 * R_2^1$ thus by rewriting the formula, R_2^1 can be obtained by: $R_2^1 = (R_1^0)^{-1} * R_2^0$. In the code the transpose function was used, since the inverse of a rotation matrix is the transpose of the matrix, while the transpose is a lighter calculation than the inverse, thus reducing the stress on the hardware.

After each rotation matrix relative to the previous sensor was calculated, the angles could be obtained. These calculations were different for each sensor, since the order of the Euler angles were different for each sensor. The first set was YXY from R_2^1, secondly ZY from R_3^2 and lastly ZX rotations from R_4^3. the following formulas were used to obtain the angles dependent on the joint set, with respect to the element of the rotation matrix:

For the first set, YXY:

$y_0 = atan2\left(R_2^1(0,1), R_2^1(2,1)\right)$ $\quad x = \arccos\left(R_2^1(1,1)\right)$ $\quad y_1 = atan2(R_2^1(1,0), R_2^1(1,2))$

For the second set, ZY:

$z = atan2\left(R_3^2(1,0), R_3^2(0,0)\right)$ $\quad y = \arcsin(-R_3^2(2,0))$

For the last set, ZX:

$z = atan2\left(R_4^3(1,0), R_4^3(0,0)\right)$ $\quad x = atan2\left(R_4^3(2,1), R_4^3(1,1)\right)$

This resulted in the angles necessary for the robot arm. Note that these equations have been obtained using the Euler angle sheet from Geometric Tools. [32]

Testing

Angle conversion test

The angles generated with this method were tested using MATLAB. A known set of angles for the robot arm was taken. This set of joint angles was first used to calculate the angles that the MARG would need to have had at those angles. For this calculation the rotation matrices necessary were calculated in the software, using the angle order of the robot arm and the known rotation matrix for each angle. After the states of the MARG's were calculated, these values were used in the formula that has been described in the angle conversion node sub chapter. The resulting angles had to be the same as the starting joint angles. This test was further automated to test multiple sets of angles, which showed that a case passed or failed to generate the same output from the input.

Software testing

Software testing on the nodes has been done using unit testing and component testing. The Gtest framework has been used for unit testing. This framework is included in the ROS test package, which comes installed with the main installation of ROS. Through unit testing the separate functions of a node could be tested. Due to time constraints the focus has been on performing this test on one node, which was the angle calculation node. Component testing was also done using the ROS test package. Through component testing it was possible to conclude whether each node was running on the right frequency and publishing on the right topics.

For the unit testing a separate C++ program had to be created. For a unit test, test cases had to be created. There are several possibilities for these test cases, the most important ones are testing a function for returning true, false, or retuning an equal value to a value set in the test case. For the testing done on the angle calculation only tests for an equal result have been performed.

The component test possibilities supplied with the ROS test package can be performed by creating a small script in which the parameters for testing could be set. These parameters were the topic which needed to be tested: the expected publication frequency; the allowed error and the time that the test waits for initialization of the nodes. Similar parameters were needed for the publication testing; the only difference was that this test does not need the expected publication speed. This test uses a time in which there needs to be published on the topic once, or in which no publications are expected instead.

System validation testing

Before testing the full system, an attempt has been made to connect the system to a human arm and to a simulation of the robot arm. During this test there was concluded that the complete system was not accurate enough to be paired with the robot. The robot arm in the simulation was moving through the body of the robot (drift), although the human arm was in the rest position of the robot. Sensor data from that test was gathered to generate insights in the cause of this drift. To remove the chance that bad calibration was the cause, each sensor was individually connected, with the shortest cable, to the Raspberry Pi. After recalibration the sensor data was read, while still only having that one sensor attached. In this case the filter generated the expected orientations. An attempt on recalibration of the sensors was also performed with the longest cable. This resulted in instable orientations from the filter again.

To conclude whether the proposed system was generating angles accurate enough to move the robot arm, the accuracy had to be tested. For this purpose, the sensor system was mounted to the robot arm. The reason for this was that there is full control over the angles in which the robot arm is set with a resolution of 0.015 radians. Since the angles of the robot arm are known with a high accuracy, it was possible to calculate the accuracy of the total system based on the input angles versus the output angle set of the proposed system. The system was tested for multiple sets of pre-determined angles, which give representative positions for the human arm. The mounting places of the proposed system were similar to the places where the system was mounted on the human arm, thus one sensor on the torso; one above the elbow on the arm (after joint 3); one before the wrist (after joint 5) and one on the hand (after joint 7). Both the input angles and the output angles were saved in a text file, which was used for the accuracy calculations.

For each of the sets the accuracy of each joint was calculated, using the input from the robot arm and the output of the system. The total accuracy was also calculated using the same combinations, but overall the joints from one set of inputs and outputs.

Results

Calibration of the sensors

The calibration results showed in this section had been based on one sensor. The other sensors have been calibrated in the same way, however, since the bias and scale factors are different for each sensor, there was chosen to only include the calibration data for sensor three. The data for calibration of the sensor was collected at home.

The data collected for the calibration of the accelerometer and the gyroscope can be found in Figure 20 and Figure 21 respectively. The bias was estimated for the accelerometer and the gyroscope and has been shown in Table 7. After applying the bias to the dataset, the resulting data had been plotted again in Figure 22 for the accelerometer and Figure 23 for the gyroscope. The bias for the accelerometer was small, resulting in a small difference between the calibrated and not calibrated dataset. The gyroscope however, had larger biases that needed to be accounted for, with -0.80 degrees per second for the Z axis being the largest bias.

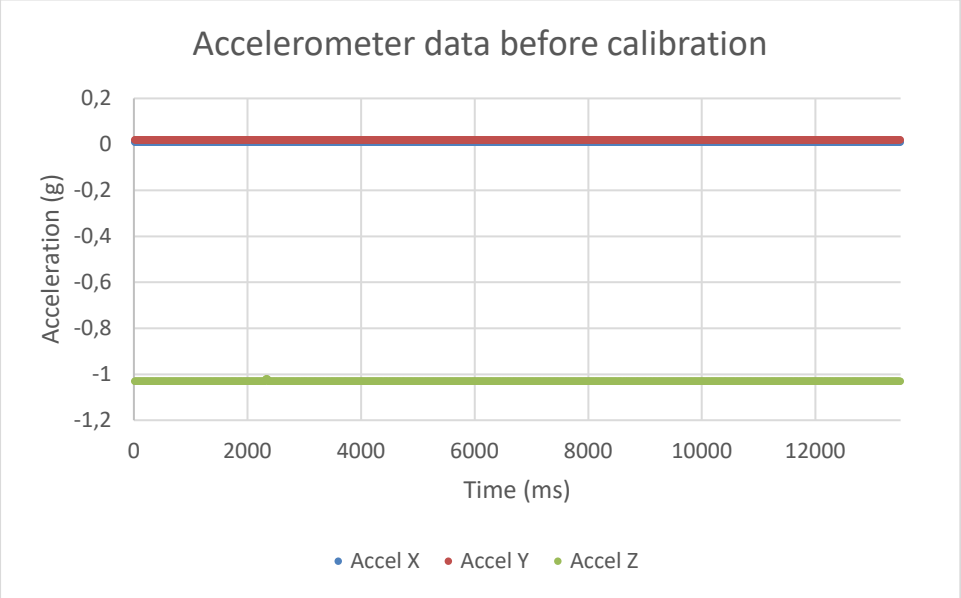


Figure 20: Accelerometer data before calibration, with the sensor placed still in one the Z axis while collecting the data.

Table 6: The calculated biases for the accelerometer and gyroscope. The bias for the accelerometer was small, while the gyroscope had a larger bias.

	Accelerometer bias (g)	*Gyroscope bias (dps)*
X axis	0.01	-0.46
Y axis	0.02	-0.36
Z axis	0.03	-0.80

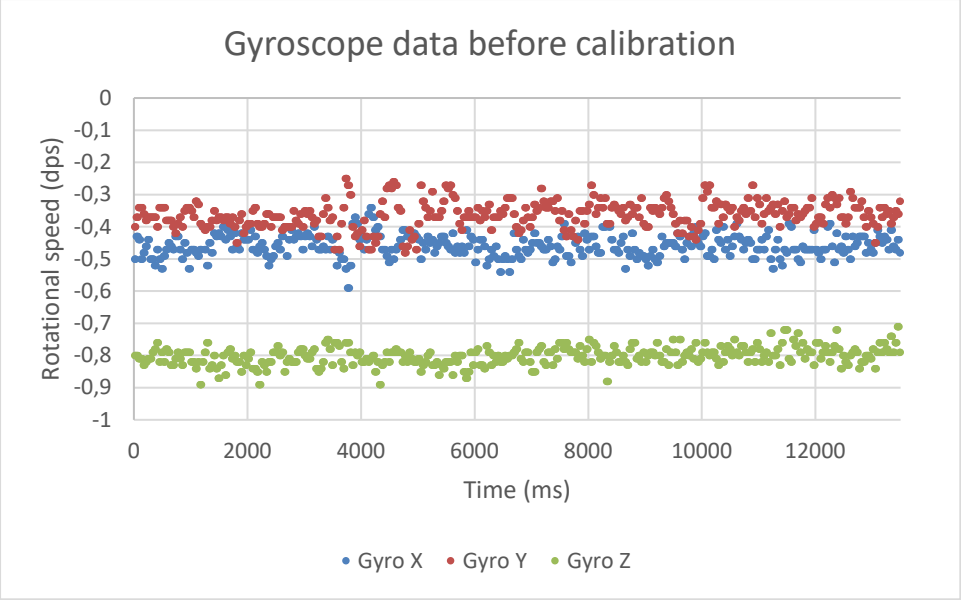


Figure 21: Gyroscope data before calibration. The sensor was held in one position while collecting the data.

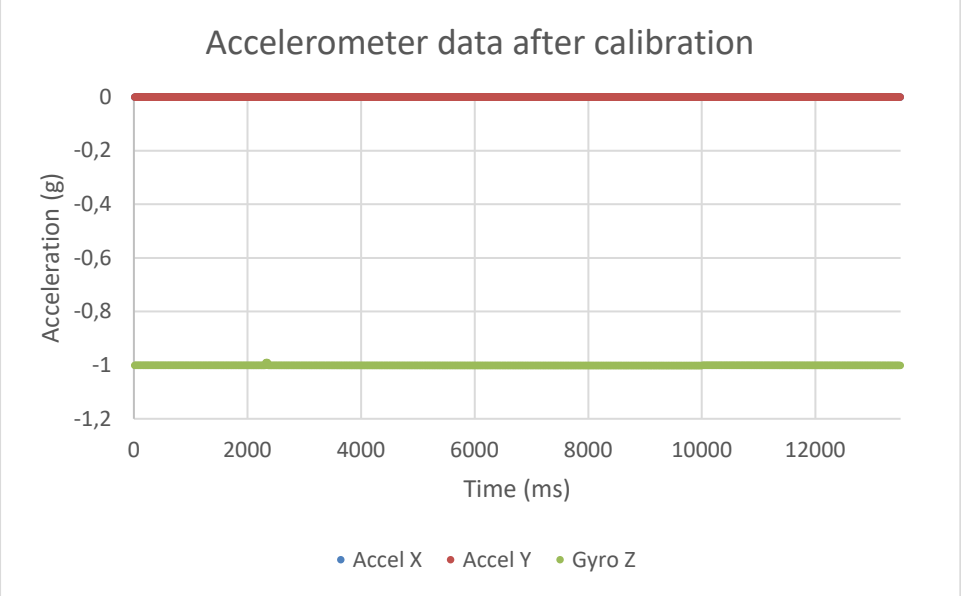


Figure 22: accelerometer data after calibration. The sensor was placed with its bottom directing towards the ground.

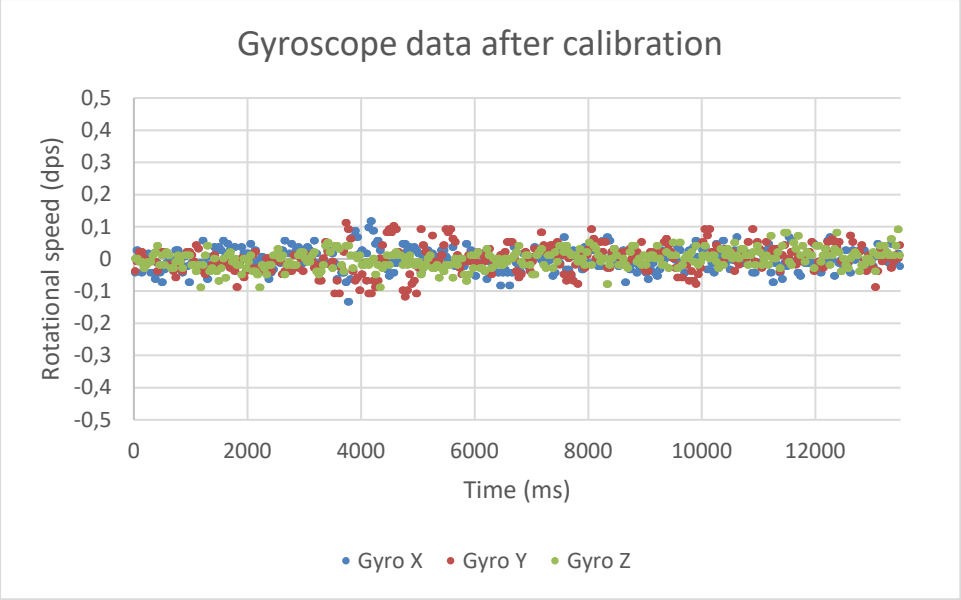


Figure 23: gyroscope data after applying the estimated biases. The gyroscope data is centered around 0 for all axis after calibration.

The data collected for the calibration of the magnetometer from this sensor has been plotted in Figure 24. Based on this data, the bias and scale factor for the sensor has been calculated. The bias and scale factor for each axis can be found in Table 8. These biases and scale factors have been applied to the data collection, which resulted in the calibrated sensor data displayed in Figure 25. After the calibration the results show that the magnetic field measured by each axis within the sensor are centred around 0, with maxima and minima around 0.5 and -0.5 Gauss respectively.

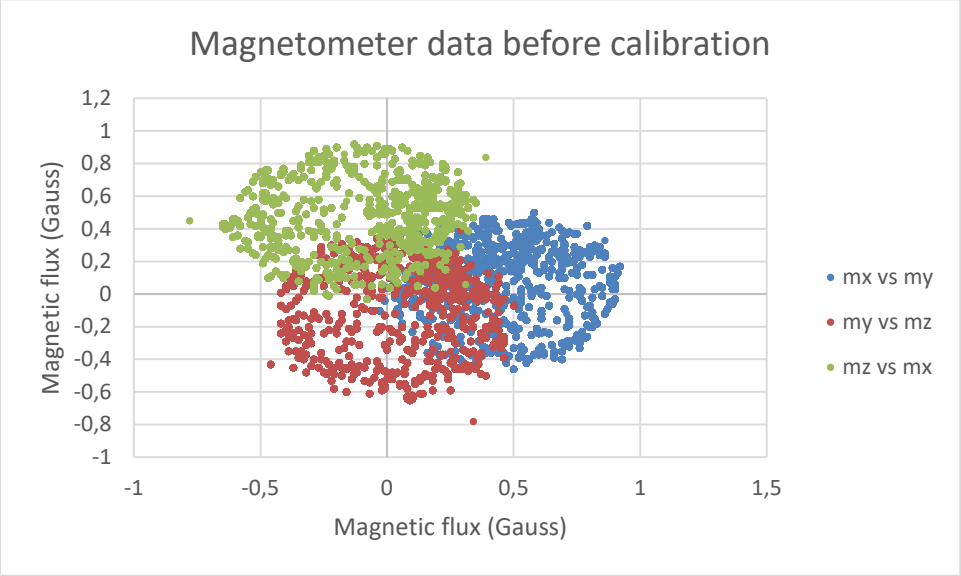


Figure 24: Magnetometer data before calibration. Each axis has a bias (offset) from the center position which makes the magnetometer inaccurate, causing this data to be unusable for the sensor data filter.

Table 7: calculated bias and scale factor based on the sensor data. The bias was subtracted from the new data and afterwards the data was multiplied with the scale factor.

	Magnetometer bias (G)	*Magnetometer scale factor*
X axis	0.445	1.081
Y axis	0.020	1.069
Z axis	-0.195	0.877

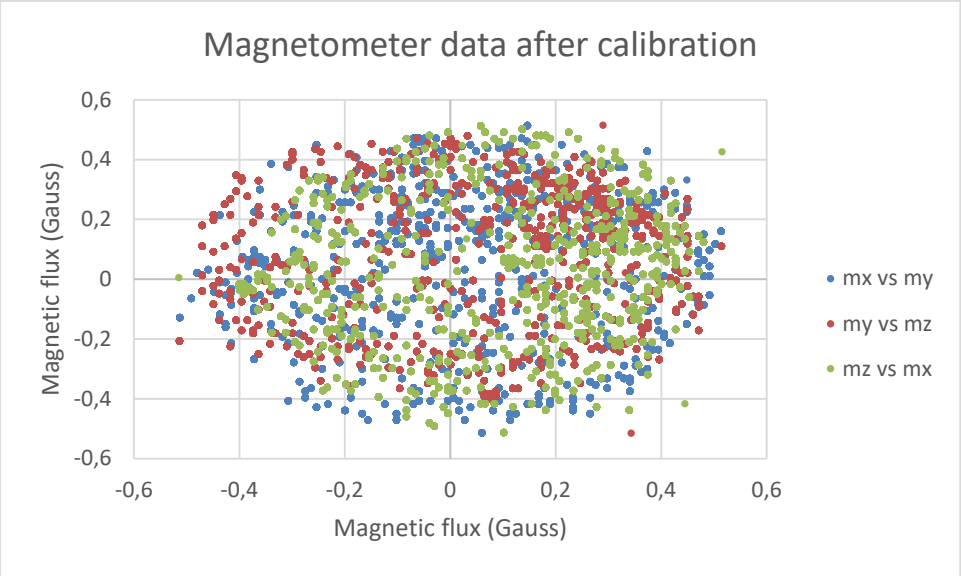


Figure 25: Data of each magnetometer axis after calibration. Each axis is centered around 0 and the maxima and minima are around +- 0.5 Gauss, which is the expected magnetic field on the surface of the earth.

Angle calculation

The seven sets of angles described in Table 9 have been tested using the MATLAB script. 6 out of 7 passed without problems. Only set 7 gave problems in backwards calculating the angles of joint 1 and 3. Instead of the rotation of 90 on joint 1 and 90 on joint 3, a rotation of 180 had been returned for joint 1 and a rotation of 0 had been returned for joint 3.

Table 8: angle sets used to test the angle calculation inside MATLAB, in degrees.

	Joint 1 (Y)	*Joint 2 (X)*	*Joint 3 (Y)*	*Joint 4 (Z)*	*Joint 5 (Y)*	*Joint 6 (Z)*	*Joint 7 (X)*
Set 1	0	0	90	45	0	45	90
Set 2	0	90	0	90	90	0	45
Set 3	0	0	90	0	0	-45	0
Set 4	0	0	0	45	90	0	-45
Set 5	0	90	90	90	45	45	90
Set 6	90	45	90	0	-90	45	0
Set 7	90	0	90	90	0	0	90

Software testing

Unit testing was only performed on the angle calculation node. The test cases have been described in Table 9. From these test cases one case failed. The case that failed was case 2 from the MARG2rotm function. this case tested the calculation of the rotation matrix. The expected matrix was an identity matrix rotated by 180

degrees over the Z axis and 90 degrees over the X axis. The returned matrix was close to the expected matrix but resulted in a .001 difference for multiple elements in the rotation matrix, were 0 was expected. Since the goal was to have equal results, this case failed.

Table 9: Unit test cases for the angle calculation node. each function is given with the description, the input and the expected output.

Function	*Test case(s)*	*Description of function*
MARG2rotm	Input 1: $\begin{matrix} X & Y & Z \\ 0 & 0 & 0 \end{matrix}$ Output 1: $R = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$ Input 2: $\begin{matrix} X & Y & Z \\ \pi/2 & 0 & \pi \end{matrix}$ Output 2: $R = \begin{matrix} -1 & -0 & 0 \\ 0 & -0 & 1 \\ 0 & 1 & 0 \end{matrix}$	Converts the Yaw, Pitch, Roll angles in radians into the rotation matrix
Calc_1_2	Input 1: $R_1^0 = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$ $R_2^0 = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$ Output 1: $R_2^1 = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$ Input 2: $R_1^0 = \begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{matrix}$ $R_2^0 = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$ Output 2: $R_2^1 = \begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{matrix}$	Takes two rotation matrices as input, uses these two matrices to calculate the angle between the two.
Angle_order	Input 1: $\begin{matrix} 0 & \pi/2 & 0 \\ \pi/2 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$ Output 1: $0\ \pi/2\ 0\ \pi/2\ 0\ 0\ 0$	Uses the angles from the backward conversion to calculate the order set for the robot arm.

Both component tests passed, with the frequency test measuring the expected 100 Hz on each topic and the publication test not receiving any message while the sensor data read node was disabled.

Validation testing

The sensor data generated with the small test on the simulation software was studied due to the inaccuracies in the movement of the visual model of the robot arm. Although the sensors were all calibrated, one of the sensors was giving a magnetic field of 0.4 Gauss on the Z axis in static position with the bottom of

the case on a table, while the other 3 were -0.3, 0.38 and 1.3 Gauss in the same orientation. Recalibrating each sensor and reading data separately with the shortest cable resulted in readings of 0.38, 0.40, 0.39 and 0.42 Gauss for the 4 sensors on the orientation used before. Changing the cable to the longer one resulted in an average measurement of 0.7 Gauss again.

Validation testing of the proposed system was planned to be executed according to the methods described in the implementation section. The test was not executed however, due to the problems found during the test with the simulation of the robot arm.

Conclusion and recommendations

It was not possible to build a working proof of concept within the timeframe of the project. The separate parts of the system were tested and working. It was possible to calibrate and read accurate data from each sensor when connected separately on a short cable. Using the longer cables, or having al sensors connected to the system resulted in some sensors that were giving inaccurate data for the magnetometer. Because this problem only occurs when the module is connected to a long cable (or multiple on the 4 combined cables), there has been concluded that there is some interference happening to the sensor data of the magnetometer on longer cable lengths. This issue can likely be solved by using premade shielded cables, reducing the chance of interference on the sensor data.

There are several improvements possible on the hardware level, to improve the overall system performance and increase the longevity of the system. First of all, the protocol used to communicate between the Raspberry Pi and the sensor modules is not ideal for the distances used. SPI is made to be on short distances, preferably a on PCB connection. For longer distances a protocol like CAN is used more commonly. Furthermore, the connectors used on the PCB's are not ideal. These connectors are surface mounted, which means that when a strong pulling force is applied (while disconnecting the wires as an example) the solder connections to the PCB could break. Since the connectors are used for the ability of quickly disconnecting the sensors, it would be better to use through hole connectors.

A better improvement might be to completely remove all cables between the sensor modules. With the power usage of current devices, capacity and sizes of rechargeable batteries and the availability of wireless communication options, it is possible to use 4 small devices that share no cable connection and connect to a host device using a protocol like Bluetooth low energy. This would make it possible to place the main controller on the robot instead of the human, reducing the components needed to be mounted on the human operator. In this case the SPI protocol for sensor connection will not be an issue either, since the controller is close to the sensor module on each of the devices.

After the calibration on short cables, the sensor data seems to be accurate. The method used for the magnetometer is however only based on the direct maximum and minimum, this does not have to be a problem, however with chances of outliers in the data it could be possible that there are mistakes in the calibration. This has not been taken into consideration during the calibration of the sensors. This also seem to have happened to the mx data used in Figure 25, since one data point has an offset from the other data points. This error could however be solved by taking the average of the lowest 1% of measurements for the minimum and the highest 1% of the measurements for the maximum. This way, the error generated by an outlier will be reduced.

With good calibration of the sensors, the accuracy of the Madgwick algorithm of 4° (claimed by K. Winer) seems to be possible. However, this conclusion has only been based on visual representation of the filter output compared to the actual sensor module, so further accuracy should be performed on the sensor data.

From the tests performed with the angle calculation there can be concluded that the proposed conversion method is valid for translating the measurements of multiple MARG's into the angles for the robot arm is valid. There are multiple ways to solve the issue regarding the measurement errors with y_0 and y_1 in the angle calculation. One of the solutions could be moving the sensors to different positions such that sensor 2 on the upper arm only measures the first two angles, which are YX, the third sensor with the following YZ and the last sensor measuring the YZX movement on the hand. This, however, makes it more difficult for

mounting the sensors on an arm, since the mounting possibilities are not ideal at the upper arm. Another possibility is by assuming that if the x rotation at that point is 0, the rotation is only done by y_1, since that is the same restriction as the human arm has, in this case mounting the sensor modules is also easier since placing the sensor near the shoulder on the upper arm could cause interference on the movements of the human. Therefore, the software has assumed that with an angle of 0 on the x axis, the rotation is performed on the y_1 axis.

Based on the results of the sperate sensor calibration; the output of the filter on well calibrated sensor data and the test performed on the angle calculation, the proposed measurement system is viable. Although it has not been possible to test the complete system, there can still be concluded based on the separate parts that a system using MARG's to track human arm movement will enable a humanoid robot to move according to the human arm. Since all the angles on the human arm are measured, full resemblance of the human arm can be achieved with this method. A completed proof of concept should still be tested in order to calculate the accuracy of the total system.

List of definitions and abbreviations

Inertial measurement unit (IMU)
: A self-contained system that measures linear and angular motion. This is usually done with a triad of gyroscopes, a triad of accelerometers. [12]

Magnetic Angular Rate and Gravity (MARG) sensors
: A self-contained system that combines the measurements done by an IMU with a triad of magnetometers. [23]

Time-of-Flight (ToF)
: Distance measurements based on the time of the emitted photons to be reflected on an object. This enables accurate distance measurements regardless of surface characteristics of an object. [33]

Micro-Electro-Mechanical systems (MEMS)
: A technology that can be defined as miniaturized mechanical and electro-mechanical elements that are made using microfabrication processes. [34]

Radio Frequency (RF)
: A radio frequency signal refers to a wireless electromagnetic signal, often used for communication. Radio waves are a form of electromagnetic radiation with identified radio frequencies. The frequency refers to the rate of oscillation. [35]

Degree of Freedom (DoF)
: The Degrees-of-Freedom refer to the specific number of axes in which a rigid body can move in a three-dimensional space. it defines the number of independent parameters of a mechanical system. [36]

Attitude heading reference system (AHRS)
: 3D orientation based on sensor data fusion from integrated gyroscope data combined with the gravitational element from the acceleration and the magnetic field from the magnetometer. This results in a drift free orientation. [37]

General purpose input/output (GPIO)
: Pins on a controller without a specific function that can be changed by software.

References

[1] H. Liu and L. Wang, "Gesture recognition for human-robot collaboration: a review," *International journal of industrial ergonomics,* vol. 68, pp. 355-367, 2018.

[2] S. S. Rautaray and A. Agrawal, "vision based hand gesture recognition for human computer interaction: a suvey," *artificial intelligence review,* vol. 43, no. 1, pp. 1-54, 2012.

[3] F. Mola, J. Antoch, L. Frigau and C. Conversano, "Classification of images background substraction in image segmentation," *Mathematica,* vol. 55, no. 1, pp. 73-86, 2016.

[4] E. Lachat, H. Macher, T. Landes and P. Grussenmeyer, "Assesment and calibration of a RGB-D camera towards a potential use for close range modeling," *remote sensing,* vol. 7, pp. 13070-13097, 2015.

[5] O. Sprdlik, "Detection and estimation of human movement using inertial sensors: applications in Neurology," Czech Technical University , Prague, 2012.

[6] google, "About Soli," [Online]. Available: https://atap.google.com/soli/. [Accessed 13 02 2019].

[7] google, "google daydream," 2018. [Online]. Available: https://vr.google.com/daydream/. [Accessed 06 02 2019].

[8] Facebook Technologies LLC., "Oculus Rift," [Online]. Available: https://www.oculus.com/rift. [Accessed 06 02 2019].

[9] TDK, "InvenSense MPU-6050," [Online]. Available: https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/. [Accessed 13 02 2019].

[10] TDK, "InvenSense MPU-9250," [Online]. Available: https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/. [Accessed 13 02 2019].

[11] Bosch sensor tec, "bosch BNO055," [Online]. Available: https://www.bosch-sensortec.com/bst/products/all_products/bno055. [Accessed 13 02 2019].

[12] Xsens, "Xsens IMU," [Online]. Available: https://www.xsens.com/tags/imu/. [Accessed 06 01 2019].

[13] UAVcoach, "How to fly a drone," [Online]. Available: https://uavcoach.com/how-to-fly-a-quadcopter-guide/. [Accessed 19 02 2019].

[14] R. Cheng, "Dig this: i operated a giant excavator from 2,500km distance," CNET, 05 03 2015. [Online]. Available: https://www.cnet.com/news/i-operated-a-giant-excavator-from-2500km-away-and-it-was-trippy/. [Accessed 19 02 2019].

[15] M. D. Do, "joystic controlled robot arm using an arduino," Instructables, 2016. [Online]. Available: https://www.instructables.com/id/Joystick-Controlled-Robot-Arm-Using-an-Arduino/. [Accessed 19 02 2019].

[16] B. Mitov, "Control of a smart robot car with a joystick," hackster.io, 29 06 2016. [Online]. Available: https://www.hackster.io/mitov/arduino-and-visuino-control-smart-car-robot-with-joystick-8a7410. [Accessed 19 02 2019].

[17] H. Singh, "Kalman Filter," Towards data science, 30 03 2018. [Online]. Available: https://towardsdatascience.com/kalman-filter-interview-bdc39f3e6cf3. [Accessed 18 02 2019].

[18] H. Singh, "Extended Kalman Filter," Towards Data Sience, 07 04 2018. [Online]. Available: https://towardsdatascience.com/extended-kalman-filter-43e52b16757d. [Accessed 18 02 2019].

[19] H. Singh, "Unscented Kalman Filter," Towards Data Science, 27 04 2018. [Online]. Available: https://towardsdatascience.com/the-unscented-kalman-filter-anything-ekf-can-do-i-can-do-it-better-ce7c773cf88d. [Accessed 18 02 2019].

[20] P. van de Maele, "Reading an IMU without the kalman filter: the complementary filter," 26 04 2013. [Online]. Available: http://www.pieter-jan.com/node/11. [Accessed 7 03 2019].

[21] altervista, "Kalman filter vs complementary filter," 25 09 2011. [Online]. Available: https://robottini.altervista.org/kalman-filter-vs-complementary-filter. [Accessed 7 03 2019].

[22] R. Mahony, T. Hamel and J. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," *IEEE Transactions on Automatic Control,* vol. 53, no. 5, pp. 1203-1218, 2008.

[23] S. O. H. Madgwick, A. J. L. Harrison and R. Viadyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, Zurich, Zwitzerland, 2011.

[24] K. Winer, "affordable 9 dof sensor fusion," github, 30 1 2016. [Online]. Available: https://github.com/kriswiner/MPU6050/wiki/Affordable-9-DoF-Sensor-Fusion. [Accessed 15 6 2019].

[25] V. Andrearczyk and P. F. Whelan, "Deep learning in texture analysis and its application to tissue image classification," in *Biomedical texture analysis*, Dublin, Vision systems group, 2017, pp. 95-129.

[26] Robot operating system, "Why ROS," Open Source Robotics Foundation, [Online]. Available: http://www.ros.org/is-ros-for-me/. [Accessed 18 02 2019].

[27] Open Source Robotics Foundation, "ROS wiki, ROS concepts," Open Source Robotics Foundation, [Online]. Available: http://wiki.ros.org/ROS/Concepts. [Accessed 18 02 2019].

[28] J. Deber, R. Jota, C. Forlines and D. Wigdor, "How much faster is fast enough? User perception of latency & latency improvements in direct and indirect touch," in *Proceedings of the 33rd Annual {ACM} Conference on Human Factors in Computing Systems* , Seoul, 2015.

[29] STC Admin, "V-Model," Software testing class, 29 05 2012. [Online]. Available: https://www.softwaretestingclass.com/v-model/. [Accessed 21 02 2019].

[30] Open source robotics foundation, "install ros kinetic ubuntu," [Online]. Available: http://wiki.ros.org/kinetic/Installation/Ubuntu. [Accessed 21 05 2019].

[31] wiringPi, "download and install wiringPi," [Online]. Available: http://wiringpi.com/download-and-install/. [Accessed 21 05 2019].

[32] D. Eberly, "Euler angle formulas," Geometric Tools, Mountain View, 1999.

[33] STMicroelectronics, "proximity sensors," 04 2018. [Online]. Available: https://www.st.com/en/imaging-and-photonics-solutions/proximity-sensors.html?querycriteria=productId=SC1934. [Accessed 06 02 2019].

[34] MEMS and Nanotechnology Exchange, "About MEMS," [Online]. Available: https://www.mems-exchange.org/MEMS/what-is.html. [Accessed 06 02 2019].

[35] Mouser electronics, "RF wireless technology," [Online]. Available: https://eu.mouser.com/applications/rf-wireless-technology/. [Accessed 18 02 2019].

[36] techopedia, "Degrees of Freedom," [Online]. Available: https://www.techopedia.com/definition/12702/six-degrees-of-freedom-6dof. [Accessed 18 02 2019].

[37] XSENS, "xsens AHRS," [Online]. Available: https://www.xsens.com/tags/ahrs/. [Accessed 29 04 2019].

[38] D. Xia, C. Yu and L. Kong, "The development of micromachined gyroscope structure and circuitry technology," *Sensors,* vol. 14, pp. 1394 - 1473, 2014.

[39] H. Lee and S. Jung, "Gyro sensor drift compensation by kalman filter to control a mobile inverted pendulum robot system," in *2009 IEEE international conference on industrial technology*, Deajoen, 2009.

[40] F. Scholcover and D. J. Gillan, "Using temporal sensitivity to predict performance under latency in teleoperation," *Human factors: The journal of the human factors and ergonomics society,* vol. 60, no. 1, pp. 80-91, 2017.

[41] G. Anderson, R. Doherty and S. Ganapathy, "User perception of touch screen latency," 2011.

[42] N. Beckers, "The human controller. manual control and the crossover model.," university of twente, Enschede, 2016.

[43] Leap Motion Inc., "Leap Motion technology," 2018. [Online]. Available: https://www.leapmotion.com/technology/. [Accessed 06 02 2019].

[44] MI.MU, "MI.MU glove technology," make a spectacle, [Online]. Available: https://mimugloves.com/tech/. [Accessed 27 02 2019].

[45] Mathworks, "auto tresholding canny edge detection," MathWorks, 2017. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/61237-auto-thresholding-canny-edge-detection. [Accessed 27 02 2019].

[46] solenerotech, "kalman filter," wordpress, [Online]. Available: https://solenerotech1.wordpress.com/2013/10/10/how-to-combine-data-from-gyro-and-accel/. [Accessed 27 02 2019].

[47] L. Skovsgaard, "zacobria universal robots," Zacobria Pte. Ltd. [Online]. [Accessed 1 03 2019].

[48] W. de Souza, "Sensor fusion algorithms for autonomous driving," medium, 13 06 2017. [Online]. Available: https://medium.com/@wilburdes/sensor-fusion-algorithms-for-autonomous-driving-part-1-the-kalman-filter-and-extended-kalman-a4eab8a833dd. [Accessed 11 03 2019].

Appendices

Appendix A – measurement techniques of the sensors inside IMU's and MARG's

Accelerometers are sensors dedicated to measure linear acceleration. The sensor usually consists of a mass that is connected to the sensor's platform using a spring-dampener link. As the sensor accelerates, the mass will move proportionally to the opposite direction of the movement within the sensor housing, on one axis. The displacement of the mass can be measured. Capacity measurement of the displacement is often used and transformed to the sensor output. Most of the accelerometers are created using MEMS fabrication to be more efficient (several μA). For MARG's a triad of accelerometers (one per axis) is used. Accelerometers face issues regarding their precision and accuracy. Most of the output errors are caused by time drift, gain and bias errors, which depend on temperature. Other errors are misalignment of sensor axis and zero mean noise. [5]

Gyroscopes measure the angular rate of change. In MEMS sensors the most common type of gyroscope is based on the Coriolis effect. A proof mass is supported by two springs and two dampeners equivalently. When the x-axis is used as the driving direction another axis will be the sensing direction. When the proof mass works under simple harmonic vibration by applying an electrostatic, piezoelectric, electromagnetic or electrothermal force, the x-axis will displace. This displacement causes vibrations on the sensing axis, caused by the Coriolis force. The angular rate of the driving direction can be calculated from the measurements of the sensing direction. The measured data is the acceleration of the vibrations. The data is then integrated over time to generate the angular rate. This also causes the drift in these measurements. The principle of the Coriolis effect is schematically represented in Figure 26. Gyroscopes suffer from similar errors as accelerometers, but has additional errors caused by non-linearity and non-orthogonality. [5, 38]

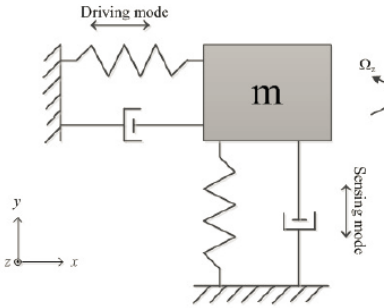


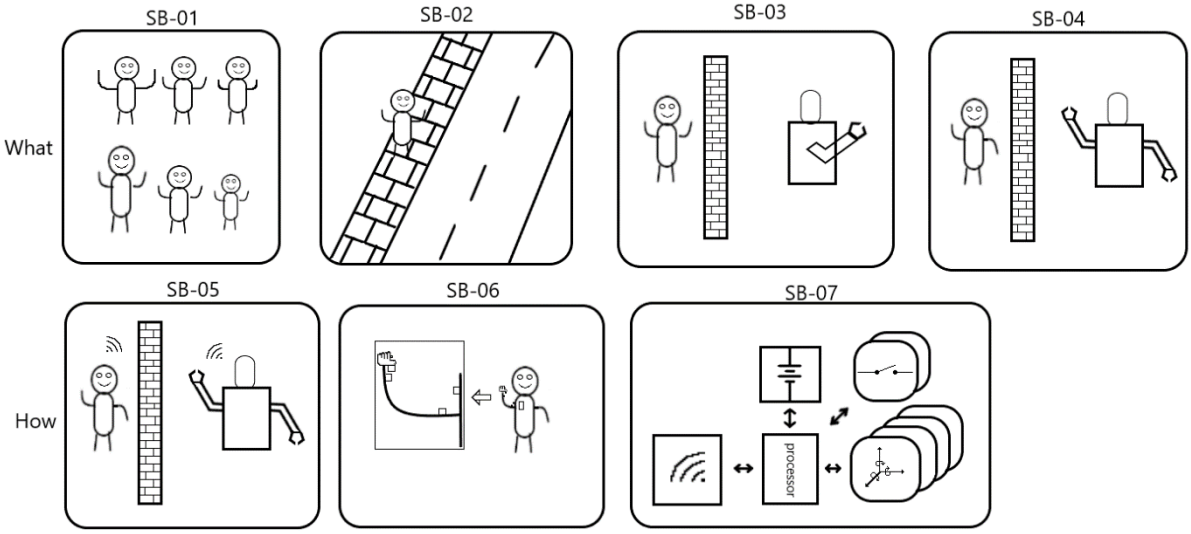
Figure 26: MEMS Gyro based on the Coriolis effect. When the sensor vibrates on the driving axis (X) the measurements will be performed on the sensing axis (Y). [7]

In calculations, linear equations are often preferred over non-linear functions. This is caused by the fact that linear systems are easier to do calculations on. However, some sensors (and therefore systems) are non-linear, which is caused by the fact that they use a different driving direction than the direction in which measurements are done. The gyroscope is an example of these non-linear sensors. The data that they generate are based on cosine or sine, which are non-linear. Orthogonality of sensors is the property that means changing A does not change B. Since gyroscopes measure in a different direction than their movement is, these sensors have a non-orthogonal behaviour. The drift that the gyroscopes suffer from is an accumulated error created by the integration which is done to obtain the angles based on the acceleration of the sensor. This generates a change in the output, while the sensor did not move [39].

Magneto meters measure the intensity and direction of the magnetic field. When the magnetic field of the earth is strong enough these magneto meters can be used as a compass, the normal strength of the magnetic field on the surface of the earth is roughly 0.5 Gauss.

Appendix B – requirements of remote arm control system

Story boards



A system that will enable the robot to be controlled wirelessly, by mimicking the movement of a human operator.

User stories

ID	US-01	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-01	Priority	Must
User story			
As a system demonstrator, I want to be able to use the system on users with different arm lengths, to increase the number of people able to operate the system such that there will be a larger interest for the system during demonstrations.			
Extra rationale			
Everyone has different lengths of their arms. A system that can adapt for these differences is able to be used on more operators.			
Acceptance criteria			
Anyone shall be able to use the system regardless of the length of their arms.			

ID	US-02	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-02	Priority	Must
User story			
As a demonstrate, I want people to learn to operate the system fast, without knowledge about the system so I can have more people experience moving the robot arm.			
Extra rationale			
To increase the number of people able to operate the system, someone with little knowledge about the system should be able to use the system without long training. Since movement of the arm of a human is natural this time could become low.			
Acceptance criteria			
A random user shall be able to operate the system within an average time of **<TBD>** minutes.			

ID	US-03	Creation date	11-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-03, SB-04, SB-05	Priority	Must
User story			
As an operator, I do not want to be restrained in my movement so that I can move around a room and move my arm without cables limiting my movement.			
Extra rationale			
Cables will hinder the free movement and cause the operator to be stuck within limited range from the robot. This means that the robot can only operate at a short range, and not with a wall or building between the robot and the operator. Short cables can also limit the amount of movement of the arm.			
Acceptance criteria			
The robot and the system shall communicate wirelessly. The system shall be battery powered. The placement of the sensors shall not limit the operator in the movements.			

ID	US-04	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-04, SB-05,	Priority	Must
User story			
As a user I want to have the robot mimic my arm movement so that I can move my arm naturally.			
Extra rationale			
Remote control of a robot with common systems requires movement that is not natural for humans. Using a system that mimics movement of an operator's arm, the operator can move a robot with the same movements as he would naturally perform			
Acceptance criteria			
The system shall mimic the operator's movements.			

ID	US-05	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-04	Priority	Must
User story			
as an operator I do not want to get frustrated by using the system so that I am willing to use the system.			
Extra rationale			
System latencies can cause user perception of the system to be frustrating. When an operator gets frustrated the operator will prefer not using the system.			
Acceptance criteria			
Latency and movement range should be in an acceptable **<TBD>** region for the user.			

ID	US-06	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-06, SB-07	Priority	Must
User story			
As an operator I want the system to be safe so that I can trust that I can use the system without being endangered, since the system has to be placed on my body.			
Extra rationale			
The system will be attached to the body of the user. The user wants to know that the system is safe before using the system.			
Acceptance criteria			
The system shall work according to the low voltage directive. Furthermore, the battery shall be protected using a battery management system.			

ID	US-07	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-07	Priority	Must
User story			
As a (daily) operator, I want to be able to temporarily turn of the movement of the robot so that I can keep wearing the sensors while temporarily doing something else.			
Extra rationale			
Since users need to wear the system it will be easier when the user can temporarily disable the system. This way the user does not need to completely take the sensors of when there happens to be a problem with the system.			
Acceptance criteria			
The system shall contain a switch to temporarily enable or disable the movement of the arm.			

ID	US-08	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-06, SB-07	Priority	Must
User story			
As an operator I want to control the different joints and the gripper of the robot arm so that I can move the robot arm to its full extend.			
Extra rationale			
Acceptance criteria			
The system shall have the sensors needed to measure the parameters needed to control the arm and gripper of the robot			

ID	US-09	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-06	Priority	Must
User story			
As an operator I want to be able disconnect the sensors separately so that I can remove the sensors easier after usage.			
Extra rationale			
Acceptance criteria			
The sensors shall have separate connectors to make wire connections between them			

ID	US-10	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked Storyboard	SB-06, SB-07	Priority	Must
User story			
As a system demonstrator I want that the system is not based on camera's so that I can use the system in more environments such that there are less problems regarding lightning conditions			
Extra rationale			
Acceptance criteria			
The system shall not use cameras to mimic the movement of a human arm.			

User requirements

Number	Requirement	Priority	Source
U01	The system shall enable the robot to mimic human motion	Must	C. Schipper
U02	The system shall have sensors needed to measure the parameters that are needed to control the robot arm.	Must	C. Schipper
U03	The system shall have sensors needed to measure the parameters that are needed to control the robot gripper	Must	C. Schipper
U04	The system shall not use cameras to mimic the movement of a human arm.	Must Not	C. Schipper
U05	The system shall operate on a battery	Must	C. Schipper
U06	The battery shall have a battery management system	Must	C. Schipper
U07	The system shall work according to the low voltage directive	Must	C. Schipper
U08	The robot and the system shall communicate wirelessly	Must	C. Schipper
U09	The placement of the sensors shall not limit the operator in its movement	Must	C. Schipper
U10	The system shall be able to be disabled temporarily	Must	C. Schipper
U11	The system shall contain connectors at each sensor for easy removal after usage	Must	C. Schipper
U12	Anyone regardless of arm length shall be able to use the system.	Could	C. Schipper
U13	A random user shall be able to operate the system within an average time of **<TBD>** minutes.	Could	C. Schipper
U14	Latency and movement range shall be in an acceptable **<TBD>** region for the user.	Could	C. Schipper

ID	REQ_US-12	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-01	Priority	Must
Requirement			
Anyone regardless of arm length shall be able to use the system.			
Rationale			

ID	REQ_US-13	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-02	Priority	Must
Requirement			
A random user shall be able to operate the system within an average time of 5 minutes.			
Rationale			
5 minutes is a good time for demonstrating purposes. This time allows people to experience the arm movement. If the time is longer, there are less people interested in the experience.			

ID	REQ_US-09	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-03	Priority	Must
Requirement			
The placement of the sensors shall not limit the operator in its movement			
Rationale			

ID	REQ_US-08	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-03	Priority	Must
Requirement			
The robot and the system shall communicate wirelessly			
Rationale			

ID	REQ_US-05	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-03	Priority	Must
Requirement			
The system shall operate on a battery			
Rationale			

ID	REQ_US-14	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-05	Priority	Must
Requirement			
Latency and movement range should be in an acceptable **<TBD>** region for the user.			
Rationale			

ID	REQ_US-07	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-06	Priority	Must
Requirement			
The system shall work according to the low voltage directive			
Rationale			

ID	REQ_US-06	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-06	Priority	Must
Requirement			
The battery shall have a battery management system			
Rationale			

ID	REQ_US-10	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-07	Priority	Must
Requirement			
The system shall be able to be disabled temporarily using a switch			
Rationale			

ID	REQ_US-01	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-04	Priority	Must
Requirement			
The system shall mimic the operator's movements.			
Rationale			

ID	REQ_US-02	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-08	Priority	Must
Requirement			
The system shall have the sensors needed to measure the parameters needed to control the arm of the robot			
Rationale			

ID	REQ_US-04	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-10	Priority	Must
Requirement			
The system shall not use cameras to mimic the movement of a human arm.			
Rationale			

ID	REQ_US-11	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-09	Priority	Must
Requirement			
The sensors shall have separate connectors to make wire connections between them.			
Rationale			

ID	REQ_US-03	Creation date	12-02-2019
Status	Proposed	Creator	Colin Schipper
Linked User story	US-08	Priority	Must
Requirement			
The system shall be able to be disabled temporarily using a switch			
Rationale			
The system shall have the sensors needed to measure the parameters needed to control the gripper of the robot			

Functional requirements

Number	Requirement	Priority	Source
SF1	The system shall enable the robot to mimic human arm motion	Must	C. Schipper
SF1.1	The system shall produce the parameters needed for movement of the robot arm and gripper	Must	C. Schipper
SF1.1.1	The system shall measure the angles of the human joints, which it converts to the angles for the robot arm.	Must	C. Schipper
SF1.1.1.1	The system uses 4 accelerometers for movement of the robot arm	Must	C. Schipper
SF1.1.2	The gripper from the robot must be open or closed	Must	C. Schipper
SF1.1.2.1	The system uses a button for movement of the robot's gripper	Must	C. Schipper
SF1.2	The system shall give accurate movement in all lighting conditions	Must	C. Schipper
SF1.2.1	The system shall not use cameras to mimic a human arm.	Must Not	C. Schipper
SF2	The system shall be battery powered.	Must	C. Schipper
SF2.1	The battery shall be protected using a battery management system.	Must	C. Schipper
SF3	The system shall communicate wirelessly with the robot	Must	C. Schipper
SF3.1	The robot and the system shall be separate systems that communicate with each other	Must	C. Schipper
SF3.1.1	The system shall use the wireless protocol that is already in use for the robot	Must	C. Schipper
SF4	The system must be able to be disabled temporarily.	Should	C. Schipper
SF4.1	The system shall have a switch for temporal disablement	Should	
SF5	Placement of the sensors shall not limit the movement of the operator	Should	C. Schipper
SF6	The system shall be able to put on and taken off the operator easily	Should	C. Schipper
SF6.1	The system will contain connectors at each sensor for easy connection and removal of wires	Should	C. Schipper

Non-functional requirements

Number	Requirement	Priority	Source
NF1	The system must comply with the low voltage directive	Must	C. Schipper
NF2	The latency must be lower than 300ms	Could	C. Schipper
NF3	A new user must be able to learn how to operate the system within <TBD>	Could	C. Schipper

Appendix C – System latency literature

Research into the perceptiveness of users of the new system will only be done when there is enough time remaining within the time that is given for the research. Existing literature on latency is however considered, since there are limits to the acceptable latency of a system. Using this acceptable limit, the system can be checked in overall viability.

The time delay in a system from input to the resulting output is called the latency of a system. Latency is a major cause within perceptiveness of a system. If a touch screen is pressed by a user the user expects a fast response, so in an ideal system the latency is zero. However, in real systems there is always latency, since sensors need to be read, the sensor data must be transferred, the data needs to be processed and an actuator must be used to respond on the input. It is understood that latency affects the perceptiveness of a virtual environment for a human user. [28]

Within interaction between a user and a remote-controlled robot there are several steps in which latency increases. The operator first decides on an action which he gives as input to the controller, this must then be processed by the controller. The controller translates this to the output and relays the information back to the user. Depending on the distance of the user to the system, the transmission latency can either be negligible on small distances or can influence the system when the robot is in another country than the operator. This leads to a move and wait approach in most of these distant situations, which increases the completion time and decreases the performance of these distant approaches. [40]

Humans can extrapolate motion to bridge the temporal gap between input, the feedback monitoring and adjustment of a system. The perceptiveness to latency depends on how well the operator can estimate the time spent at the velocity. An operator who is insensitive to these increases will overestimate the time needed for the movement of an end effector, while an operator who is oversensitive to these increases will underestimate the time needed for the movement of the end effector [40]. This causes unwanted behaviour since this causes the end effector to move further than wanted or not far enough, possibly resulting in systems that are not predictable, which is a major issue when dealing with robots or other moving parts from machines.

To overcome latency issues research has been done in predictive movement. With predictive movement a simulated image will be created based on the input that is given to the system. Using such systems enables an operator to mentally travel to a future position which is equal in time to the latency, however there are often mismatches between the expected position and the actual position causing the operator to use the move and wait approach [40].

To test operators on their perception of latency, different latency levels (400, 600, 800 and 1000ms) were tested on a remote-controlled car. The research was done to gather insights in the errors created using the different latency times. This research led to the conclusion that with increasing latency the time to complete a course and the amount of errors taken increased. The mean completion time went from 105.64s at 400ms latency to 181.10s at 1000ms latency. [40]

The study that was conducted by Anderson et al. determined the level of delay that is acceptable for users. A latency above 580ms was considered unacceptable to the users, however it was noted that a high latency might be tolerable for longer tasks since the tests conducted were based on small tasks [41]. Other research concluded that dependant on the task, a user could detect dragging on a touch screen at 11ms while for tapping on a touch screen the latency was perceptible at 69ms. [28]

There are guidelines which can be used to determine the maximum latency of systems as well. These guidelines have been established using the crossover model in closed-loop control of a system. Movement detection using MARG's for robotic arm movement are a 0^{th} order system. This results in a system latency

between 150ms and 300ms for acceptable error margins for usage of these systems. [42] This means that the system should be able to comply with these guidelines to make a feasible system.

Appendix D – MARG sensor choice

The basic sensor characteristics of the 3 options can also be found in Table 10. Table 11, Table 12 and Table 13 show the extended characteristics per type within the different MARG's. The most important characteristics have been placed in the decision matrix in Table 14. Some of these characteristics such as the price and the availability of libraries have been weighted more heavily, since there is limited time and budget during the project. This leads to the MPU-9250 being the best sensor between the three.

Table 10: basic sensor characteristics. The characteristics have been obtained from basic google searches and the data sheets of the sensors.

	MPU 9250	**LSM9DS1**	**BNO055**
price	€ 8,50	€ 17,95	€ 36,95
Communication protocols	I^2C SPI	I^2C SPI	I^2C UART
Addresses on I^2C	2	1	2
Libraries for raspberry pi	I^2C SPI	I^2C SPI	UART
Operating voltage	3v3	3v3	3v3

Table 11: typical characteristics of the accelerometers in different MARG's. The displayed data has been obtained from the data sheets and converted to the same units.

Accelerometer	**MPU 9250**	**LSM9DS1**	**BNO055**	**units**
ADC	16	16	14	bits
Data frequency	4000	952	100	Hz
Sensitivity per range scale	2g: 0.061 4g: 0.122 8g: 0.244 16g: 0.488	2g: 0.061 4g: 0.122 8g: 0.244 16g: 0.732	2g: 0.244 4g: 0.488 8g: 0.977 16g: 1.953	mg
Sensitivity tolerance	+- 3		+-1	%
Sensitivity temperature drift	+-0.026		+-0.03	%/ Celsius
Zero g calibration tolerance	X, Y: 60 Z: 80	X, Y, Z: 90	80	mg
Zero g temperature drift	+- 1.5		+-1	mg/ Celsius
Non-linearity	+- 0.5		+- 0.5	%
Cross axis sensitivity	+- 2		+- 1	%
Output noise	300		150	µg/√Hz

Table 12:typical characteristics of the gyroscopes in different MARG's. The displayed data has been obtained from the data sheets and converted to the same units.

Gyroscope	**MPU 9250**	**LSM9DS1**	**BNO055**	**units**
ADC	16	16	16	bits
Data Frequency	1000	952	100	Hz
Sensitivity per range scale	250 dps: 7.63 500 dps: 15.26 1000 dps: 30.77 2000 dps: 60.98	245 dps: 8.75 500 dps: 17.50 2000 dps: 70.00	125 dps: 3.81 250 dps: 7.63 500 dps: 15.26 1000 dps: 30.77 2000 dps: 60.98	mdps
Sensitivity tolerance	+- 3		+- 1	%
Sensitivity temperature drift	+- 0.03		+- 0.03	% / K
Zero dps calibration tolerance	+- 5	+- 30	+- 1	dps
Zero dps temperature drift	+- .22		+- 0.015	dps/ K
Non-linearity	+- 0.1		+- 0.05	%
Cross axis sensitivity	+- 2		+- 1	%
Output noise	+- 0.1		+-0.1	Dps

Table 13: typical characteristics of the magnetometers in different MARG's. The displayed data has been obtained from the data sheets and converted to the same units.

Magnetometer	MPU 9250	LSM9DS1	BNO055	Units
ADC	14			bits
Data frequency[1]	100	100	20	Hz
Magnetic field range	X, Y, Z: +- 4800		X, Y: 1300 Z: 2500	µT
sensitivity	0.6	400 µT: 0.014 800 µT: 0.028 1200 µT: 0.056 1600 µT: 0.112	0.3	µT
Sensitivity temperature drift			+- 0.01	%/K
Zero field offset	+- 75	+- 100	+- 40	µT
Zero offset temperature drift			+- 0.23	µT/K

1. Magnetometer data can be used multiple times between new readings, based on the assumption that the change in the magnetic field of the earth is small.

Table 14: decision matrix for the sensor modules from different manufacturers. This matrix has been made based on the details in the other tables.

	MPU9250	LSM9DS1	BNO055
Sensitivity	++	++	[]
Accuracy	+	[]	+
Drift	[]	<>	+
Price	++	[]	--
Impact on other parts of the system			
Available libraries	++	++	--
Logic level	+	+	+

++ best; + better than average; [] average; - worse than average; -- worst; <> no data

Appendix E – Module design for remote arm control system

System enable switch

Input data	Sensor data	I/O logic levels	WiringPi, gpio library for raspberry pi
Output data	System status	I/O logic levels	Publish state on topic
Application	Get switch state Debounce state change Change published state	At 100 Hz 5 or 10 cycles	

the system enable switch will be used to note whether the robot arm needs to move or whether the robot arm should stop moving. When the state of this switch is 0, the robot arm should get a command to move to the rest position. In the other case, the robot arm should move according to the movement of the human arm. the state of the switch will be published on the switch enable topic.

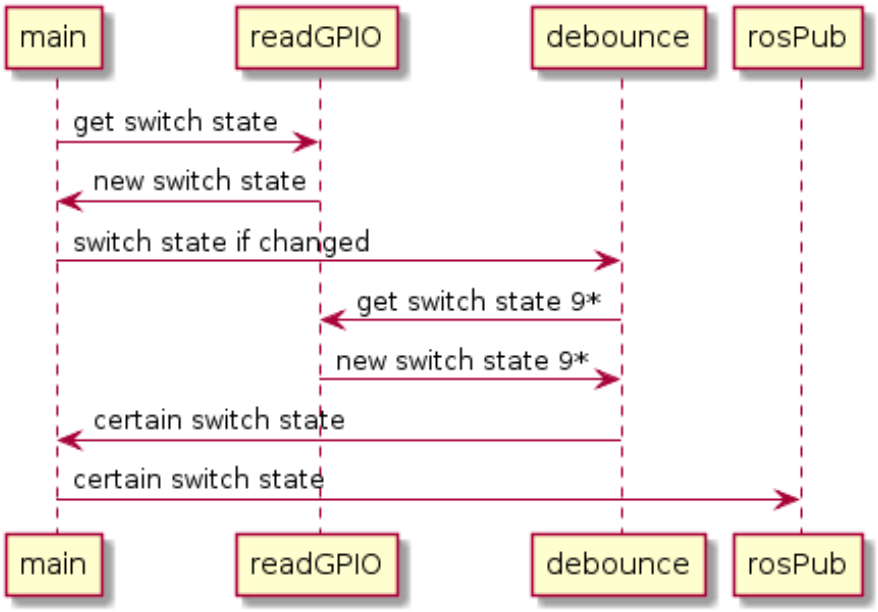


Figure 27: U.ml sequence diagram for the system enable switch node

IMU sensor data read

Input data	IMU sensor data Switch state	Accelerometer: g Gyroscorpe: dps Magneto meter: G I/O logic levels	wiringPi (gpio library) MPU9250 SPI(ported version of Bolder Flight systems Arduino library)
Output data	IMU data array	Array of IMU data for 4 sensors	Publish 4* [ax, ay, az, gx, gy, gz, mx, my, mz]
Application	Set sensor to be read Read sensor data	Set Cs pin 100 Hz	Operating speed of robot

this node uses the sensors and the switch state as input. The SPI protocol is used to communicate with the sensors. Each sensor has its own chip select (Cs) pin. When a sensor is to be read from, the chip select of that sensor will be set low. This activates the sensor such that the data can be obtained. Obtaining the data will be handled by a combination of the wiringPi library and the MPU9250 SPI library. The MPU9250 SPI library contains the addresses of the sensors from which the sensor data can be read. After the sensor data is obtained, this library also converts the data to the units needed for the AHRS algorithm. The wiringPi library handles the pin control for the raspberry pi. After the data for the 4 sensors is obtained, the data is combined into one array and published on the “multi_IMU_data” topic. The switch state is forwarded to the filter node, since the system needs to give the arm the right commands when the system is disabled.

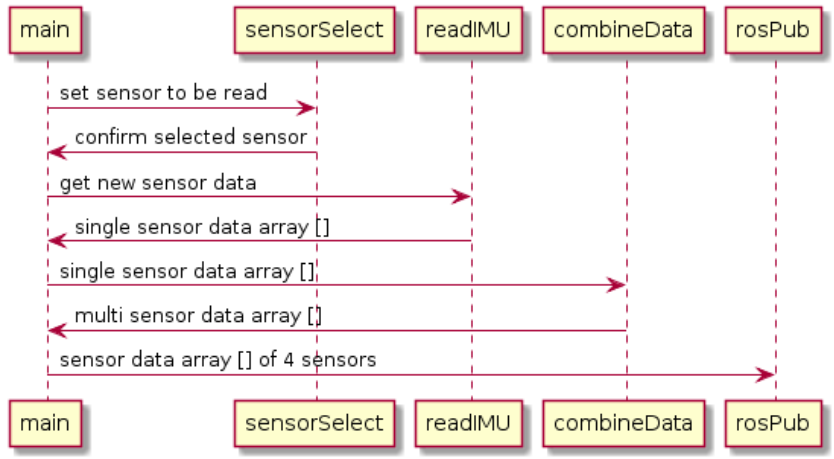


Figure 28: U.ml diagram for the sensor data read node

Sensor data fusion algorithm

Input data	IMU data	4* [ax,ay,az,gx,gy,gz,mx,my,mz]	MadgwickAHRS (filter library)
Output data	Absolute orientation	4* [R,P,Y]	
application	Split data array's Pass data trough filter		

The sensor data filter node is based on an application of the filter presented in the paper of Madgwick et all. The node uses the IMU sensor data array as input. There is a separate instance of the filter for every sensor, which updates every time that new sensor data arrives. This filter works using a gradient decent algorithm that combines the gyroscope data with the gravitational force of the accelerometer and the magnetic field measured by the magnetometer. After the filter calculated the absolute orientation for the 4 sensors, the orientations are published as an array.

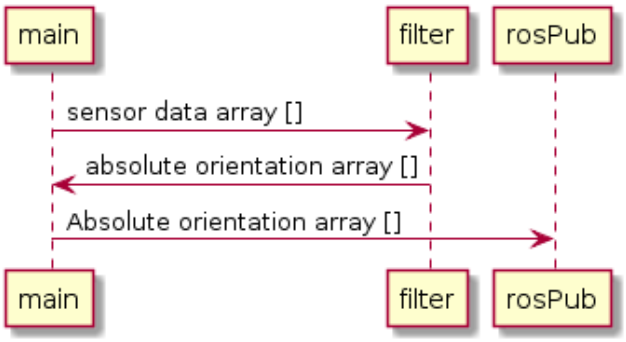


Figure 29: U.ml sequence diagram for the sensor data filter node

Angle calculation for SPERA arm

Input data	Absolute orientation	4* [R, P, Y]	Eigen 3 matrix library
Output data	Spera arm angles	Q [Y, X, Y, Z, Y, Z, X]	
Application	Convert orientation to rotation matrix Calculate relative orientation Convert relative orientation to angles	Rotm_0_2 Rotm_1_2 Q [Y, X, Y]	 First angle set for Spera

The angle calculation node uses the absolute orientation arrays as input. This data is then split into the separate sensors, after which the data gets transformed into the rotation matrices relative to the earth. The rotation matrices relative to the earth are then transformed to the rotation matrices relative to the previous body. The angles of the joints are then calculated based on the rotation matrices relative to the previous body.

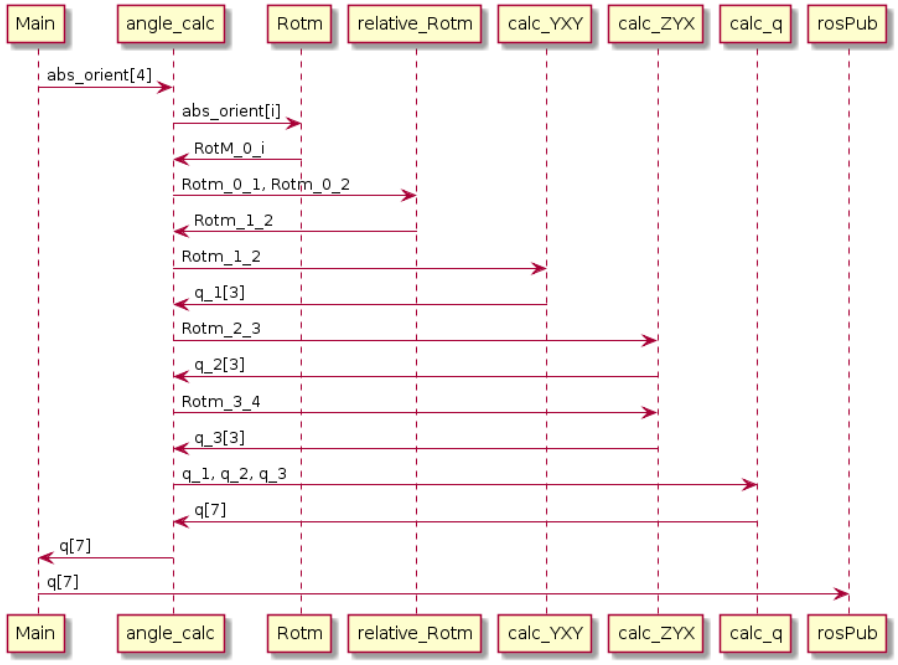


Figure 30: U.ml sequence diagram for the angle calculation node

Button read for gripper state

Input data	Sensor data Switch state	I/O logic levels I/O logic levels	wiringPi gpio library
Output data	Gripper state	‘o’ or ‘c’ dependent on state	
Application	Read GPIO pin Debounce state change Publish gripper state	100 Hz 10 cycles Press < 2s, ‘o’. press >= 2s, ‘c’	

The gripper state node will measure the state of a button, when the state of the button changes, this change is debounced to prevent multiple changes occurring within a short time. When the changed state is kept for at least 10 cycles, the time of the press is recorded, such that a press shorter than 2 seconds results in an open gripper and a press longer than 2 seconds will result in a closed gripper. After the button is released, the change is debounced again to prevent the same behavior at release. The new state is then published onto the “gripper_state” topic.

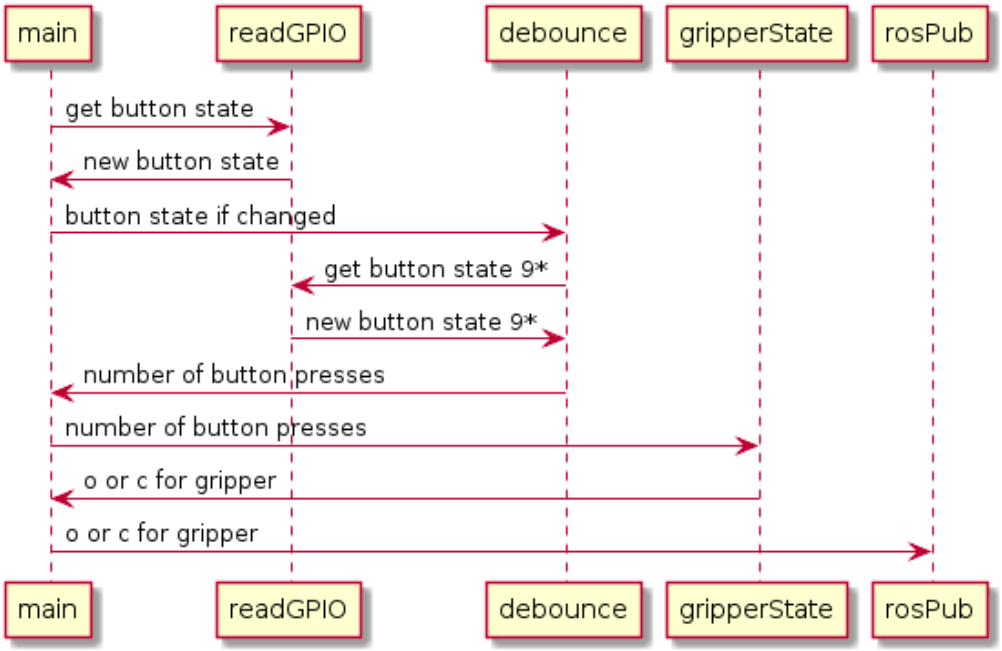


Figure 31: U.ml sequence diagram for the gripper state node