FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Autonomous navigation

Position tracking of a remote control vehicle using IMU

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Electrical Engineering and Information Technology

by

Dimitar Naydenov
Registration Number 0926254

to the Faculty of Informatics
at the Vienna University of Technology
Advisor: Dipl.-Ing. Dr.tech. Markus Bader

Vienna, 17th February, 2016

Dimitar Naydenov | Markus Bader

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Autonomes Fahren

Positionsbestimmung eines Modellbaufahrzeuges mittels IMU

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Elektrotechnik und Informationstechnik

eingereicht von

Dimitar Naydenov
Matrikelnummer 0926254

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Dipl.-Ing. Dr.tech. Markus Bader

Wien, 17. Februar 2016

Dimitar Naydenov	Markus Bader

Contents

Abstract

Autonomous robot navigation has become important for the scientific community due to the increasing interest in self-driving vehicles. In order to navigate in its environment, a robot needs to perform first an estimation of its current position, so that it could plan its route and follow it accordingly. There are several approaches to accomplish that: one example is the Simultaneous Localisation And Mapping (SLAM) technique, where a comprehensive scan of the surroundings of the robot is performed, so that an obstacle map with the position of the robot in it could be created.

In this thesis a position tracking system for detecting the coordinates of a mobile station is developed to support the method of SLAM. It relies on combining data from the input throttle and steering angle with measurements from two separate sensors, using an Extended Kalman Filter (EKF) as a correction mechanism. The sensor measurements rely on detecting the orientation of the mobile base with an Inertial Measurement Unit (IMU) and determining the travelled distance by estimating the displacement of the robot relative to the ground beneath it with the help of a laser mouse sensor. This approach has been implemented and evaluated, achieving accuracy of 3% in the estimation of the robot's coordinates for distances up to 10m on a flat surface.

Abstrakt

Die autonome Roboternavigation spielt aufgrund der zunehmenden Interesse an selbstfahrenden Fahrzeugen eine immer wichtigere Rolle in der Forschung. Eine Voraussetzung für die erfolgreiche Navigation ist die Fähigkeit des Roboters seine Position zu bestimmen. Diese kann für die weitere Planung und Ausführung einer Route verwendet werden. Es gibt unterschiedliche Methoden, um das zu erreichen: eine solche Technik ist das SLAM-Verfahren (englisch "Simultaneous Localisation And Mapping"), bei dem eine Abtastung der Umgebung des Roboters durchgeführt wird, so dass eine Hinderniskarte mit der Position des Roboters darin erzeugt werden kann.

In dieser Arbeit wurde ein Positionsermittelungssystem zur Erfassung der Koordinaten einer Mobilstation entwickelt, um die SLAM-Technik zu unterstützen. Es beruht auf Kombinieren der Daten aus der Eingabegeschwindigkeit und Lenkwinkel mit Messungen von zwei verschiedenen Sensoren mit Hilfe des Erweiterten Kalman Filters (EKF). Die Sensormessungen bestehen aus Messen der Orientierung einer mobilen Basis mit einem Inertialsensor (englisch IMU, Inertial Measurement Unit) und Bestimmen der zurückgelegten Distanz durch Abschätzen der Verschiebung des Roboters gegenüber dem Boden mit Hilfe eines Laser-Maus-Sensors. Dieser Ansatz wurde durchgeführt und evaluiert, wobei eine Genauigkeit von 3% bei der Schätzung der Koordinaten des Roboters für Entfernungen bis zu 10 Metern auf einer ebenen Fläche erreicht wurde.

1. Introduction

1.1. Problem statement

The development of a precise localisation system is critical for many robot implementations. Accurate calculation of the position and orientation of an autonomous system is necessary, so that it could navigate in its environment. Determining the location of a robot is, however, not a simple task - each measurement of vehicle displacement is relative to a certain frame, so that the selection of appropriate reference points is an issue of great importance. Several position tracking techniques include:

1. **Absolute measurement:** This approach utilises an absolute coordinate system, to which all measurements are being referred. An example is the Global Positioning System (GPS), which has been the preferred method for localisation of transportation vehicles for the last decade. A receiver mounted on the vehicle obtains signals from geostationary satellites, which provide the geographical coordinates of the object with accuracy up to 3 meters [1]. This method, however, has limited functionality indoors and lacks the precision, when it comes to short distance movements.
2. **Dead reckoning:** The method of dead reckoning relies on calculating the postition of a mobile base from its previous position, based on information about the current velocity and heading [2]. For example, measuring the rotation of the vehicle wheels could be used to calculate the travelled distance [3]. Another approach in this category is the implementation of an Inertial Measurement Unit (IMU) [4]. This is a sensor consisting of an accelerometer, a gyroscope, and sometimes a magnetometer. It can measure acceleration, angular velocity, and the direction and magnitude of the Earth magnetic field to produce information about its movement in space. Of course, precise measurements are required for this approach, because an error would grow proportionally to the travelled distance [2].
3. **Simultaneous Localisation and Mapping (SLAM):** In this approach the robot environment is scanned, most commonly with a laser rangefinder, and an obstacle map of the vehicle surroundings is generated [5]. This technique provides information of the robot's location in regard to the nearby objects it detects. This approach offers position estimation with multiple reference points. Some of the disadvantages are the complexity of the system, the increase in the required computational power, and the high price of the scanning device.

Another important subject is data fusion - the position information, obtained by several different mechanisms, needs to be combined in a certain manner, so that a single output is produced. In the course of this project, different techniques were combined and a prototype localisation system for determining the position and orientation of a mobile station was designed, built, and tested.

1.2. Methods

In this project a specialised system for determining the coordinates and orientation of a mobile station relative to a starting position is developed. The system is based on the principle of dead reckoning, meaning that the position of the robot is calculated using data from its previous location. The system is mounted on a remote controlled car for test purposes. It relies on a combination of several position information sources. By distributing the data acquisition over multiple systems, separate reference points are utilised for location estimation, which increases the redundancy and improves the reliability of the position tracking. The developed system uses the following set of data sources, software tools, and algorithms to estimate the robot position:

1. **Mechanics model of the robot:** the input throttle and steering angle together with mathematical representation of the vehicle geometry provide basic motion information.
2. **Inertial Measurement Unit:** the IMU is provides orientation information, which is independent from the robot platform and the control commands.
3. **Laser sensor from an optical mouse:** the mouse sensor measures the dispacement of the vehicle relative to the ground beneath it.
4. **Robotic Operating System (ROS):** a software framework, designed to provide a universal platform for the development of diverse robot applications.
5. **Extended Kalman Filter:** the filter uses the data acquired from the mechanics model and the sensors to produce a position estimate, based on the confidence interval, attributed to each data source.

An onboard microcontroller is used to configure the sensors. It reads the data from both sensors and then transfers it to a communication module, which provides connection to a remote computer station over WiFi. The microcontroller receives control commands over the same wireless connection to operate the throttle and the steering of the vehicle. After data acquisition a sensor fusion algorithm - the Extended Kalman Filter - is performed onboard the computer station. The filter weighs the input data according to a predefined set of measurements, in order to produce a more accurate state estimate.

Since none of the original components of the chassis have been modified or replaced, the system could be potentially used on any ground robot platform. From hardware perspective, difficulties arise from the proper alignment of the IMU sensor and from the secure mounting of the laser sensor close to the ground. In order to function correctly, the software requires a relatively precise mathematical representation of the model mechanics, consequently an adequate correction mechanism, optimised for the platform in use could be applied.

The system was tested on several separate tracks to produce a comprehensive analysis of the obtained results.

1.3. Evaluation platform

For the purpose of this project, a radio controlled car is used as a test platform. The position tracking equipment is mounted on a RC vehicle, operated remotely by a computer. Thus, the properties of the system could be easily demonstrated through a series of tests. By estimating the performance in a simulation environment, an appropriate assessment of the system's capabilities could be obtained.

The RC model used in this project is based on Tamiya TT-02 four-wheel drive chassis. It features a 540Nm DC motor, controlled by an Electronic Speed Controller (ESC), a 2.4GHz radio module, consisting of a transmitter and a receiver, and a servo for controlling the steering of the vehicle. Power is delivered by a 7.8V, 1500mAh NiMH battery pack.

The relatively large chassis of the vehicle offers a convenient platform for mounting the position tracking system components. The powerful motor allows testing the performance of the system at high velocity. The high-capacity battery provides sufficient power supply for both the vehicle and the tracking system.

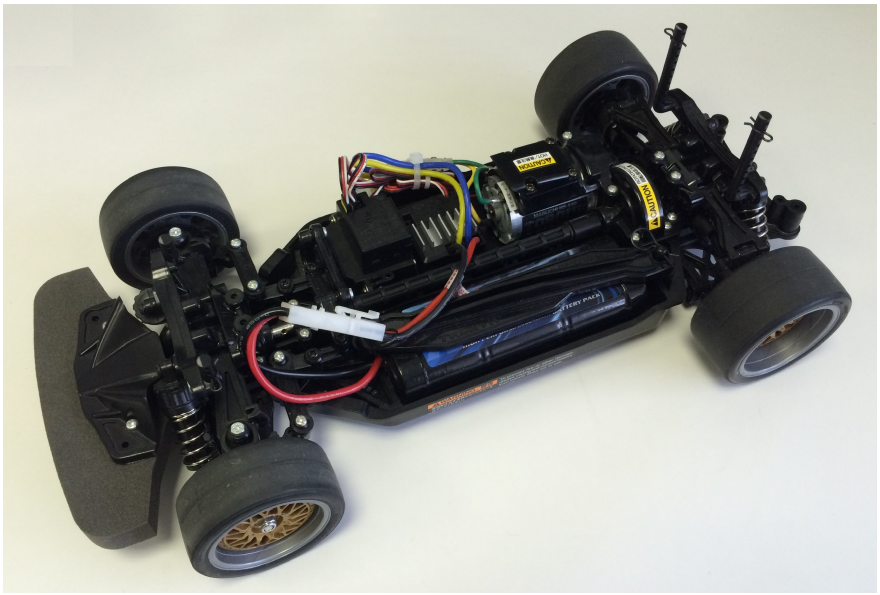


Figure 1.1.: Tamiya TT-02 chassis

Control mechanism:

The transmitter sends control signals over the 2.4GHz radio link to the receiver, using two different channels for the motor and for the steering. A pulse-width-modulation (PWM) technique is used for operating the ESC and the servo. In this modulation scheme the control signal consists of periodic rectangular pulses with a specific duration. The speed and the angle of steering can be controlled by modifying the pulse length.

Such setup enables to redirect the input signal for the ESC and servo from the radio receiver to a microcontroller, capable of generating this type of pulses. Thus, the vehicle can be operated from a remote computer station and the original radio unit can be used as backup control mechanism.

Mechanics model:

The chassis of the Tamiya model represents a geometry, known as Ackermann steering: as figure 1.2 shows, it is characterised by the difference in the steering angles ϕ_{in} and ϕ_{out} for the inner and outer wheels of the vehicle. The difference minimises the side slip of the wheels, caused by the different radius of rotation r_{in} and r_{out}. By approximating the model to a tricycle with a steering wheel positioned in the middle of the front axle, the two different steering angles can be combined into a single one ϕ [6]. The control input in this configuration comprises of a single linear velocity $\vec{v}$ and a single steering angle ϕ.

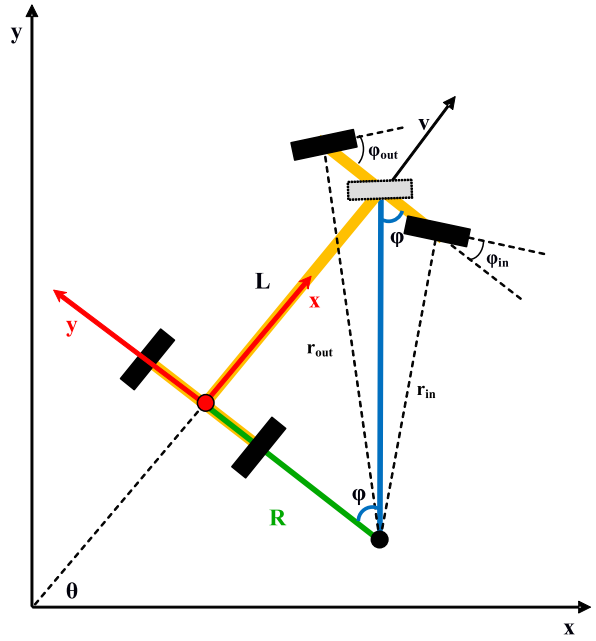


Figure 1.2.: Ackermann steering geometry used for the project

A series of tests are performed in order to establish the relationship between the input PWM-values and the corresponding linear velocity $\vec{v}$ and steering angle ϕ. The resulting model is used to provide a rough estimation of the current position and orientation of the mobile base. According to the geometry, an angular velocity could be defined as the product between the current velocity and the momentary radius, which is a function of the steeing angle ϕ:

$$tan(\phi) = \frac{L}{R} \qquad R = \frac{L}{tan(\phi)} \qquad \omega = \frac{v}{R} \qquad \omega = \frac{v \cdot tan(\phi)}{L} \tag{1.1}$$

The system state (x, y, θ) could be described in this case by the following equation:

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} + v_t \cdot dt \cdot cos(\theta_{t-1}) \\ y_{t-1} + v_t \cdot dt \cdot sin(\theta_{t-1}) \\ \theta_{t-1} + \omega_t \cdot dt \end{pmatrix} = \begin{pmatrix} x_{t-1} + v_t \cdot dt \cdot cos(\theta_{t-1}) \\ y_{t-1} + v_t \cdot dt \cdot sin(\theta_{t-1}) \\ \theta_{t-1} + \frac{v_t \cdot tan(\phi_t)}{L} \cdot dt \end{pmatrix} \tag{1.2}$$

Of course, one has to account for the possible errors in the model construction. For example, the conversion from PWM-values to speed and angle values is prone to measurement errors. Moreover, the inertia of the vehicle has to be taken into consideration because the robot needs time to accelerate from $\vec{v_1}$ to a certain velocity $\vec{v_2}$. To get better estimates a set of sensors and a correction algorithm are used to improve the accuracy in the position tracking of the vehicle.

2. Setup

2.1. Hardware Configuration

This section provides a detailed description of the postition tracking system mounted on the test vehicle. Besides an overview of the used components, it illustrates the principle of operation of the selected hardware, as well as the communication to a remote computer station.

2.1.1. Component overview

The main controller, used in the project, is an ATMega328P, which is mounted on an Arduino Nano development board. It is connected to the MPU-6050 IMU module over an I2C interface for gathering the orientation data. The displacement of the vehicle is measured by an ADNS-9500 laser sensor and is sent over a SPI interface to the Arduino board. The microcontroller transfers the sensor data together with a timestamp to a communication module, the ESP-8266, using a serial interface. The ESP-8266 acts as a WiFi access point, to which a remote computer connects. The module sends the data, received over the serial link, as IP packets for further analysis. Using the same TCP link control data is sent to the Arduino, which drives two PWM-pins to operate the ESC and the servo of the vehicle. The whole system is powered by the car battery.

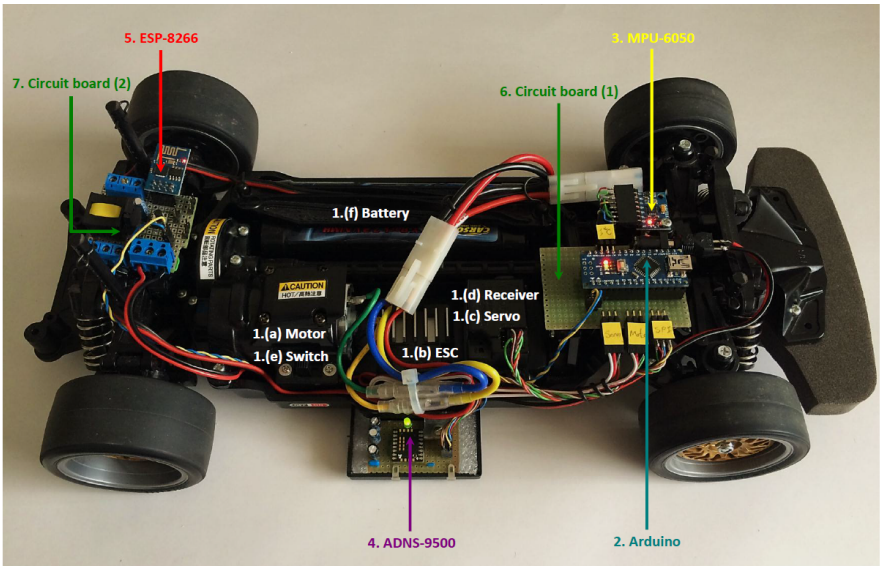


Figure 2.1.: Position tracking system mounted on a Tamiya TT-02 Chassis

The correction in the position of the vehicle takes place onboard the remote computer, which uses the data from the mechanics model and the sensors as inputs to an Extended Kalman Filter (EKF), which then produces the corrected state. The separation of data acquisition and algorithm

execution on different platforms allows a high sensor sampling frequency of 100Hz, which minimises the discretisation error.

The system components were installed without much effort on the chassis, using the available screws, securing the different parts of the RC vehicle. The IMU was aligned with the axes of the vehicle to simplify the setup and the laser sensor was mounted close to the surface in order to capture the ground images, necessary for the displacement calculation. The alignment of both sensors is not compulsory, but allows a simple deployment of the system. A software calibration could be performed to compensate any offset in the orientation and position of the sensors relative to the chassis.

Components description

1. **Tamiya TT-02 chassis:**
   a) 540Nm DC motor: a powerful DC motor to drive the RC vehicle
   b) ESC: a PWM-controlled Electronic Speed Controller
   c) Servo: a PWM-controlled servo
   d) 2.4GHz Radio Receiver: used in combination with the original radio control
   e) Power switch: turns on or off the power supply to the ESC and the radio receiver
   f) Battery: a 7.8V NiMH battery
2. **Arduino Nano:** A development board for the ATMega328P microcontroller. The controller runs at 16MHz, has 32kB of onboard flash memory and 2kB of SRAM. It reads the information from the sensors and sends it to the communication module. It generates the PWM pulses, which control the speed and steering of the vehicle, based on the information received from the main computer system.
3. **MPU-6050:** Mounted firmly on the Tamiya chassis and properly aligned with the axes of the vehicle, this sensor gathers information about the acceleration and angular velocity of the mobile base in its own coordinate system Σ_{Tamiya}. It is using proprietary algorithms to compute its orientation in space relative to an inertial coordinate system Σ_{world}.
4. **ADNS-9500** is a laser sensor from an optical mouse. It is fixed steadily to the chassis 2.5mm above the ground and its axes are aligned with those of the IMU. The sensor measures the displacement of the vehicle relative to its previous position and stores the data in two 16-bit registers, one for each axis. After reading the data from the registers their contents is cleared, so that each time the displacement (dx, dy) relative to the previous position is produced.
5. **ESP-8266** serves as a WiFi-to-UART interface between the computer system and the microcontroller. It creates an 802.11g access point and acts as a TCP server. The Arduino sends data about displacement and orientation over its serial interface to this module, which is then transferred to a TCP client on the main computer system.
6. **Circuit board (1):** The main controller is mounted on this circuit board. It contains additional components, such as voltage dividers for the communication interfaces, a signal detection circuit and a multiplexer for the PWM signals.
7. **Circuit board (2):** Consists of a voltage regulator circuit to power the WiFi-module and the laser sensor. A power switch turns on or off the power supply for the whole system.

Safety mechanism

The control commands can be generated either by the operator, using a joystick or by a computer algorithm designed for autonomous navigation. Since several programs running on two processors control the speed of the test subject, it is a good practice to develop a separate, hardware solution to take control over the mobile base. It is used as a backup control mechanism, if one of the programs misperforms, the connection between the computer and microcontroller is lost and in any other scenario, in which the RC model is heading at full-speed towards an obstacle. The developed backup control system uses the original radio module as shown in figure 2.2:

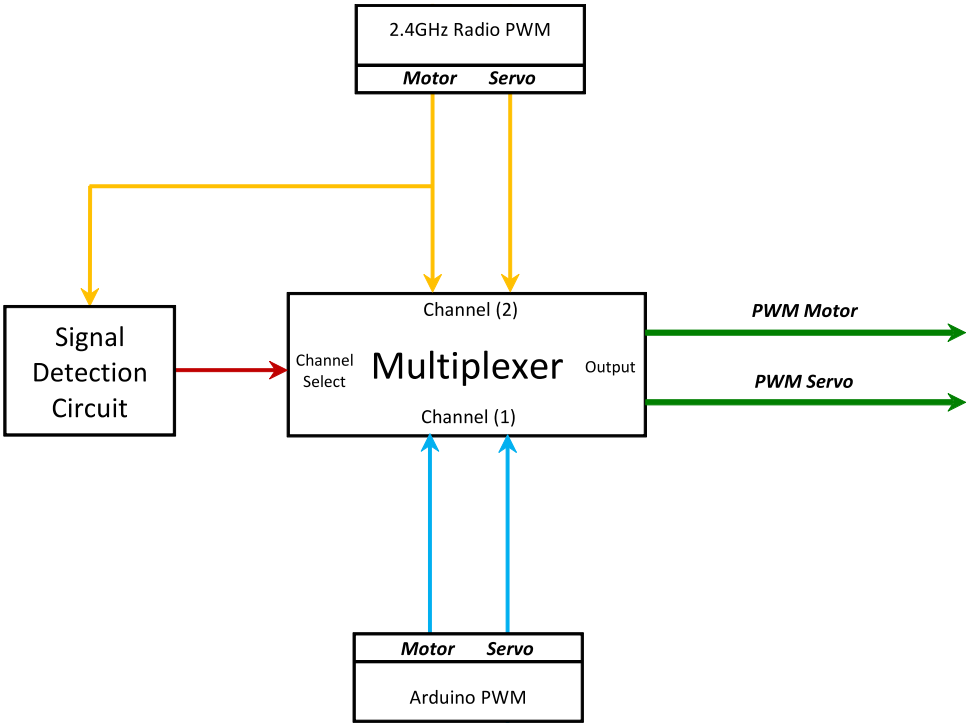
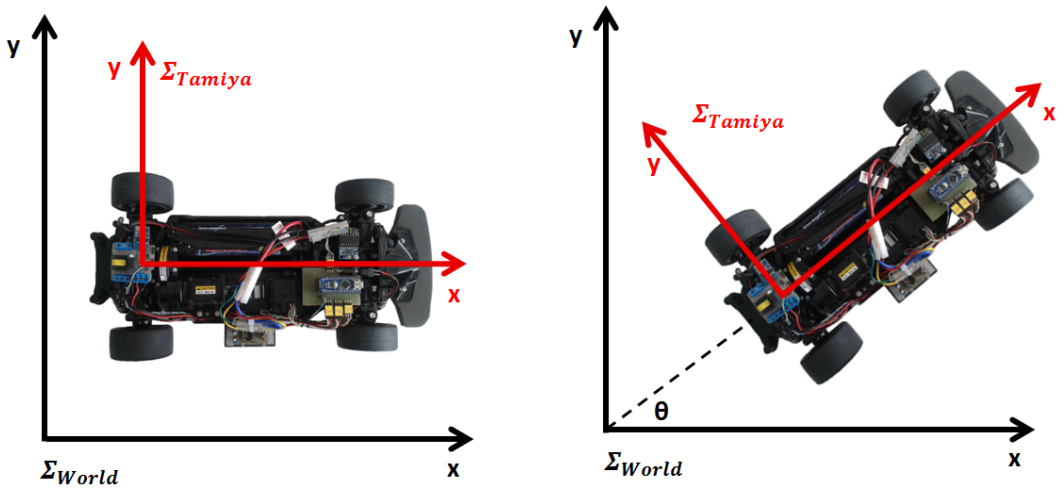


Figure 2.2.: Multiplexor circuit used for input channel selection: radio PWM signals (in yellow); Arduino PWM signals (in blue); channel select line (in red); PWM output (in green)

The PWM outputs of both the Arduino (in blue) and the radio unit (in yellow) are fed to input channels (1) and (2) of a multiplexer to enable the selection of different signal sources. The PWM motor signal line from the radio receiver is also connected to a signal detection circuit to drive the multiplexer channel select pin (in red). When the radio transmitter is switched off, zero voltage is supplied to the signal detection circuit. Thus, channel (1) is selected and the PWM signal, generated by the Arduino, is propagated to the multiplexer output (in green). When the transmitter is turned on and a PWM signal is detected, the circuit switches the input channel to channel (2), so that the PWM pulses from the radio can control the vehicle. This ensures, that even if one of the programs fails, the user can bring the mobile base to a halt or steer it to a safe location.

2.1.2. Inertial Measurement Unit

An Inertial Measurement Unit (IMU) is used for detecting the vehicle orientation in space. The IMU is a small micro-electro-mechanical sensor, consisting of an accelerometer and a gyroscope. For the purpose of this project, it is fixed firmly on the RC car, so that its x- and y-axis match those of the Tamiya chassis to measure the acceleration and the angular velocity relative to the coordinate system Σ_{Tamiya}. Since the goal of the project is to determine the position of the vehicle relative to a starting position, the data from the sensor should be transformed to represent the displacement and orientation in an inertial coordinate system Σ_{world}.



(a) Test vehicle with IMU axes aligned to Σ_{world} (b) Test vehicle with IMU rotated θ° relative to Σ_{world}

Figure 2.3.: Coordinate transformation between the local vehicle frame Σ_{Tamiya} and the global inertial frame Σ_{world}

In order to do that, an IMU with additional special characteristics was utilised. The MPU-6050 has an integrated Digital Motion Processor (DMP), which is capable of performing enhanced algorithms for calculating the orientation relative to the inertial system Σ_{world}. Taking into consideration the acceleration due to the earth gravity and constantly gathering information from the accelerometer and gyroscope, the DMP acts as a separate controller with the function of providing data about the sensor's orientation in space. Upon initialisation, the MPU-6050 interprets its current orientation as the inertial system Σ_{world} and any deviation from this initial position will be perceived as a rotation relative to the value, measured at start-up.

The MPU-6050's DMP functionality greatly reduces the load on the main microcontroller, which does not perform additional complex data fusion algorithms. The controller receives the orientation data as a quaternion $q(x, y, z, w)$ from which it calculates the angle around the z-axis.

Although the IMU can deliver the acceleration relative to the inertial system Σ_{world}, it requires a lot of effort to accurately get the travelled distance [7]. The necessary double integration of the acceleration value will drastically increase the error. Moreover, experiments have shown, that the sensor acceleration offset at rest varies with the angle, so that even the provided velocity would be inaccurate. To address this limitation a laser mouse sensor is used for measuring the displacement of the vehicle.

2.1.3. Laser mouse sensor

As previously stated, a separate sensor can be implemented to compensate for the inaccuracy of the accelerometer used for calculating the position. For this project, the ADNS-9500 laser sensor, most oftenly used in high-performance computer mice, is used for measuring the travelled distance. It is aligned to the axes of the test vehicle, so its data output is related to the Σ_{Tamiya} coordinate system. The principle of operation is the following:

A vertical-cavity surface-emitting laser (VCSEL) illuminates the ground surface with coherent light, which after reflection is focused via lens on a high-speed image acquisition system, operating at several thousand frames per second. An on-board digital signal processor (DSP) analyses and compares the taken images and determines the direction and magnitude of the movement (Moy, 2011).

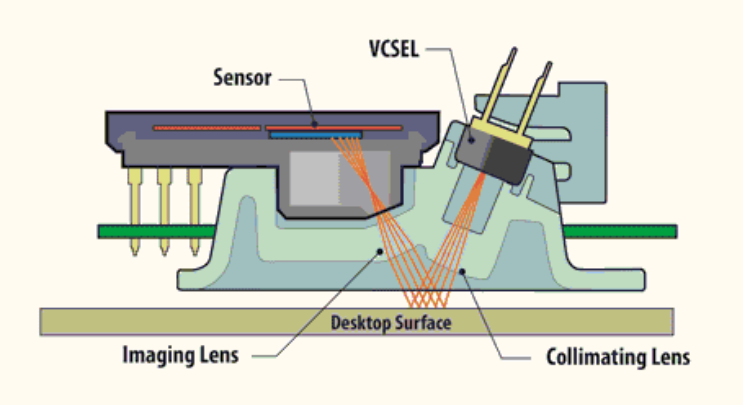


Figure 2.4.: Laser sensor[1]

The laser sensor provides the displacement Δx and Δy relative to the previous reading of its registers in form of "counts". These measured values apply in the coordinate system Σ_{Tamiya}. They have to be interpreted together with the chosen resolution, in counts per inch, or CPI, to determine the distance, which the sensor has travelled over the surface. The relationship between the value in counts and the travelled distance in meters is given in equation 2.1:

$$\Delta x[m] = \frac{\Delta x[counts] \cdot 0.0254[m/inch]}{resolution[counts/inch]} \tag{2.1}$$

Register values are reset to zero after reading, which allows one to detect differentially small displacements, when reading the registers at high enough frequencies. This feature of the module allows one to achieve relatively high accuracy in the measurements because a small change in the orientation is combined with a small change in the position.

Similar to the IMU, the main computation of the sensor data gets performed by the sensor itself - a specialised DSP with digital image processing capabilities is used for transforming the visual data into travelled distance, which offloads the microcontroller from intensive computational operations.

[1]Image obtained from Dennis Moy, *Optical mouse technology: Here to stay, still evolving*, 2011

2.1.4. Communication module

The hardware connection between the microcontroller, mounted on the vehicle, and the computer station is established by a communication module, connected to the Arduino serial interface. The ESP-8266 is a compact WiFi chip, featuring full support for the 802.11g protocol, and is capable of transmitting data it receives over its serial connection as IP packets to a remote host to provide a simple interface for connecting a microcontroller to a computer system [9]. In order to be able to test the position detection system where no connection to a wireless network is possible, the module is configured to establish an access point of its own and set up a TCP server. A client running on a laptop is set up to connect to this server and to begin the data exchange between the two devices.

The Arduino continuously sends position and orientation data, gathered from the sensors, over the serial link and the ESP-8266 forwards it as IP packets to the remote client. The same TCP socket is utilised to send control signals from the computer station to the microcontroller in order to drive the vehicle. In this case, the client sends IP packets with the necessary information to the wireless module, which then extracts the payload and forwards it over the serial link to the controller.

The wireless module offers a lot of advantages to the system:

- simple integration: the chip provides a direct communication between the computer and the microcontroller
- reliability: the TCP protocol takes care of the correct data transmission
- transparency: the wireless module hides the communication details from both the microcontroller and the computer.

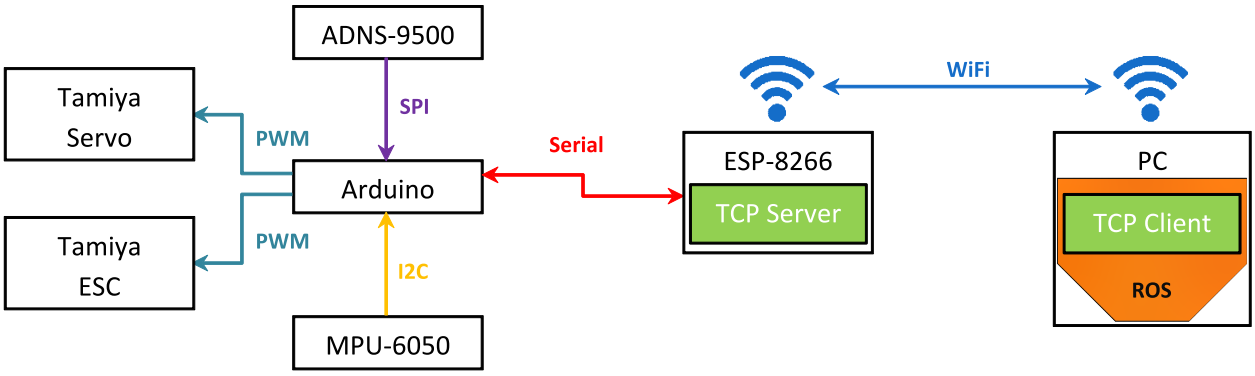


Figure 2.5.: Communication flow between the different components of the position tracking system

The interpretation of the obtained data and the control of the vehicle from a computer is made possible by using a specialised robotics software. The Robotic Operating System (ROS) is an open-source platform, which provides message passing interface for inter-process communication [10]. The main program running on the computer station, is a ROS node featuring a TCP client inside it. The position information, received from the TCP socket, is passed as a ROS-compatible message for further analysis. The ROS-network is discussed in details in the next section.

2.2. Software

This section describes the software structure implemented on the computer station and discusses in details the Extended Kalman Filter used for the data fusion.

2.2.1. ROS as middleware

The data from the sensors is sent to a computer for further analysis. This is accomplished by using the Robotic Operating System (ROS) as middleware. ROS is a collection of libraries and tools, developed for Linux and written in C++ and Python, which provide an easy to use framework to work with robots [10]. A ROS network consists of several programs, called nodes, which exchange information via publishing messages on different topics. By subscribing to a topic and by publishing information on another one, the user can easily control the data flow between the nodes. Figure 2.6 shows the network of nodes and topics, designed for the prototype position tracking system:

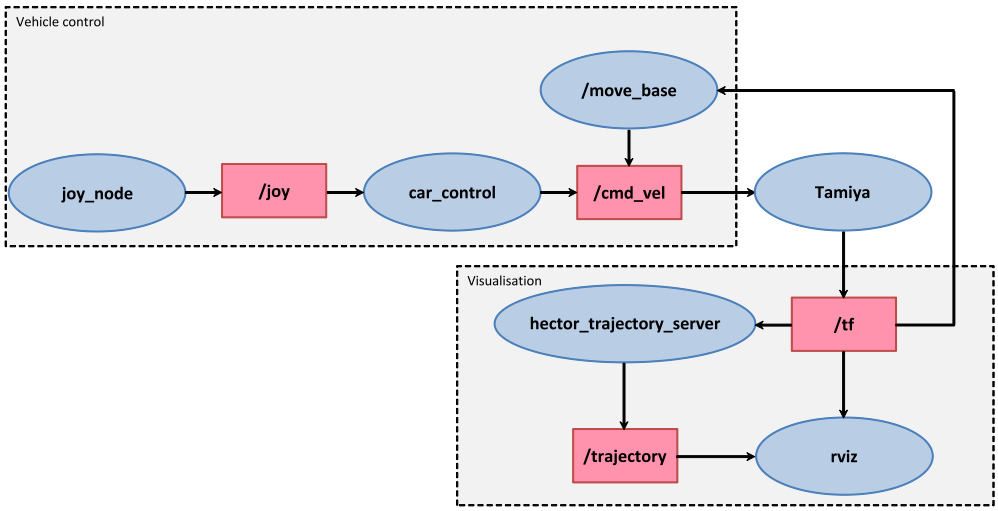


Figure 2.6.: ROS communication: nodes are coloured in blue, topics in red

On the computer, the ROS node **"Tamiya"** operates as a TCP client, which connects to the wireless module. It receives the sensor data and carries out the sensor fusion algorithm. The corrected state is published on the */tf* topic. By subscribing the visualisation node **"rviz"** to this topic, a 3-dimensional, real-time interpretation of the vehicle state is obtained. The data from the same */tf* topic is used by the **"hector_trajectory_server"** node to pass the trajectory of the mobile base to **"rviz"**.

The **"Tamiya"** node is subscribed to the */cmd_vel* topic to receive control signals to transfer to the Arduino. As figure 2.6 shows, different nodes can be registered to publish on this topic, in order to implement different control inputs. For example manual control with a joystick would require a node **"joy_node"** to read the joystick input and another one **"car_control"** to publish the data in the appropriate message format for the */cmd_vel* topic. The use of the **"move_base"** interface allows one to implement algorithms for autonomous navigation, which could be used with predefined maps. The ROS framework enables the integration of many other functions, which would further improve the robot field of operation, but are beyond the scope of this project.

2.2.2. Kalman Filter

As already discussed, a mechanism is required to combine the postition information, obtained from different data sources. In this project an Extended Kalman Filter was used for data fusion. It is a nonlinear extention to the popular Kalman Filter, which provides a weighted average, giving more weight to the estimates with higher certainty. The Kalman Filter can be described as a set of mathematical equations that provide an efficient computational (recursive) means to estimate the state of a process, in a way that minimises the mean of the squared error (Welch and Bishop, 2004). Typical uses of the filter include smoothing noisy data and providing estimates of parameters of interest. The Kalman Filter model assumes that a system evolves from a prior state $\mathbf{x}_{t-1}$ to a new state $\mathbf{x}_t$ according to the following equation:

$$\mathbf{x}_t = A_t\mathbf{x}_{t-1} + B_t\mathbf{u}_t + \mathbf{w}_t \tag{2.2}$$

The new state vector $\mathbf{x}_t$ is a linear function of the previous state $\mathbf{x}_{t-1}$, the control input $\mathbf{u}_t$ and a certain process noise $\mathbf{w}_t$. The matrix A_t represents the state transition function, which describes the effect of the system parameters at time $t-1$ on the system at time t. The influence of the input control is modelled by the input control matrix B_t. The vector $\mathbf{w}_t$ contains the process noise terms for each parameter in the state vector. The process noise is assumed to be drawn from zero mean multivariate normal distribution with covariance given by the covariance matrix Q_t.

The system is also supposed to be observable and the performed measurements are described by the following equation:

$$\mathbf{z}_t = H_t\mathbf{x}_t + \mathbf{v}_t \tag{2.3}$$

$\mathbf{z}_t$ represents the measurement vector. The matrix H_t transforms the state vector parameters into measurement domain. $\mathbf{v}_t$ is the vector containing the measurement noise terms for each observation in the measurement vector. Like the process noise, the measurement noise is assumed to be zero mean Gaussian white noise with covariance R_t.

The algorithm comprises of two steps - state prediction and measurement update. In the prediction step, an estimate of the state parameters together with an estimate of the new covariance at time t is performed. Then, equation 2.2 is used to propagate the state parameters and the process noise Q_t is included in the covariance prediction:

$$\hat{\mathbf{x}}_t = A_t\mathbf{x}_{t-1} + B_t\mathbf{u}_t \tag{2.4}$$

$$\hat{P}_t = A_tP_{t-1}A_t^T + Q_t \tag{2.5}$$

The correction stage comprises of state correction and covariance update:

$$\mathbf{x}_c = \hat{\mathbf{x}}_t + K(\mathbf{z}_t - H\hat{\mathbf{x}}_t) \tag{2.6}$$

$$P_c = (I - KH)\hat{P}_t \tag{2.7}$$

The state correction $\mathbf{x}_c$ is calculated from the state prediction $\hat{\mathbf{x}}_t$ and from the difference between the measurement $\mathbf{z}_t$ and the measurement prediction $H\hat{\mathbf{x}}_t$ from equation 2.3. The difference is multiplied by a coefficient K, known as the Kalman gain.

$$K = \hat{P}_tH^T(H\hat{P}_tH^T + R_t)^{-1} \tag{2.8}$$

The Kalman gain is a ratio between the the predicted covariance and the measurement noise. Its role and the principle of the algorithm are best demonstrated by discussing the two extreme cases:

1. In the first one the predicted covariance $\hat{P}_t$ is small compared to the measurement noise R_t, so the Kalman gain tends to zero. The term in the brackets in equation 2.6 dissapears (or has a negligible value) and the corrected state equals the predicted state, resulting in situation where the prediction is valued more than the measurements. The new covariance P_c is equal to or slightly less than equal to the predicted covariance $\hat{P}_t$, meaning that the noisy measurement has little influence on the resulting uncertainty.
2. In the second one the measurement noise R is small compared to the predicted covariance $\hat{P}_t$. The Kalman gain approaches the identity matrix, so that $\mathbf{x}_c \approx K\mathbf{z}_t$, meaning that the confidence in the measurements is greater than the confidence in the model. The corrected covariance P_c is reduced to a fraction of $\hat{P}_t$ due to the small value of $(1-KH)$, resulting from the decreased uncertainty in the measurements.

The capability to modify the influence of each data source based on the noise associated with it makes the Kalman Filter a suitable candidate for data fusion algorithm [12]. More information about the Kalman Filter algorithm can be obtained from [11]. Subsection 2.2.3 discusses in detail how the nonlinear Extended Kalman Filter was implemented in this project.

2.2.3. Extended Kalman Filter

The following paragraphs describe the state transition equations together with the different steps of the Extended Kalman Filter algorithm.

System model

The state of the vehicle can be described by its current position on the xy-plane and its angle around the z-axis. Equation 1.2 represent the state transition function $g(\mathbf{x}, \mathbf{u})$, which depends on the prior state $\mathbf{x}_{t-1}$ and on the input control $\mathbf{u}_t$. The system uncertainty is modelled by the covariance matrix P and the process noise $\mathbf{w}_t$ (comprised of the control noise, represented by the control noise matrix M).

$$\mathbf{x}_t = \underbrace{\begin{pmatrix} x_{t-1} + v_t \cdot dt \cdot cos(\theta_{t-1}) \\ y_{t-1} + v_t \cdot dt \cdot sin(\theta_{t-1}) \\ \theta_{t-1} + \frac{v_t \cdot tan(\phi_t)}{L} \cdot dt \end{pmatrix}}_{g(\mathbf{x},\mathbf{u})} + \mathbf{w}_t; \quad P = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_\theta^2 \end{pmatrix}; \quad M = \begin{pmatrix} \sigma_v^2 & \sigma_{v\phi} \\ \sigma_{\phi v} & \sigma_\phi^2 \end{pmatrix} \tag{2.9}$$

The state transition function $g(\mathbf{x}, \mathbf{u})$ is not linear. The Extended Kalman Filter is designed for nonlinear systems and operates by linearising the state transition function at current mean μ_{t-1} and covariance P_{t-1} [13], making it suitable for this project.

State prediction

The prediction step for the state is similar to the linear Kalman Filter, equation 2.4, and uses the mechanics model from equation 1.2:

$$\hat{\mathbf{x}}_t = \begin{pmatrix} \mu_{x,t-1} + v_t \cdot dt \cdot cos(\mu_{\theta,t-1}) \\ \mu_{y,t-1} + v_t \cdot dt \cdot sin(\mu_{\theta,t-1}) \\ \mu_{\theta,t-1} + \frac{v_t \cdot tan(\phi_t)}{L} \cdot dt \end{pmatrix} \tag{2.10}$$

For the covariance prediction the state transition function will be linearised at the current mean μ_{t-1} by calculating the jacobian G of $g(\mathbf{x}, \mathbf{u})$ with respect to the current state $\mathbf{x}_{t-1}$:

$$G_t = \frac{\partial g(\mu_{t-1}, \mathbf{u}_t)}{\partial \mathbf{x}_{t-1}} = \begin{pmatrix} 1 & 0 & -v_t \cdot dt \cdot sin(\theta_{t-1}) \\ 0 & 1 & v_t \cdot dt \cdot cos(\theta_{t-1}) \\ 0 & 0 & 1 \end{pmatrix} \tag{2.11}$$

With this linearised approximation of the state transition function it is possible to forward the uncertainty according to equation 2.5, where instead of the linear transition A_t a linearised version G_t for positive vehicle speed is used. As the process noise Q_t is given by the control noise M_t, the jacobian of the state transition function with respect to the conrol values V_t should be calculated. This way the uncertainty in the control is transformed into uncertainty in the state:

$$V_t = \frac{\partial g(\mu_{t-1}, \mathbf{u}_t)}{\partial \mathbf{u}_t} = \begin{pmatrix} dt \cdot cos(\theta_{t-1}) & 0 \\ dt \cdot sin(\theta_{t-1}) & 0 \\ \frac{dt \cdot tan(\phi_t)}{L} & \frac{v \cdot dt}{L \cdot cos^2(\phi_t)} \end{pmatrix} \tag{2.12}$$

The predicted covariance $\hat{P}_t$ has therefore the following form:

$$\hat{P}_t = G_t P_{t-1} G_t^T + Q_t = G_t P_{t-1} G_t^T + V M V^T \tag{2.13}$$

Measurement prediction

The next step of the algorithm requires to calculate a measurement prediction $\hat{\mathbf{z}}_t = H\hat{\mathbf{x}}_t$ from the state prediction, according to equation 2.6. The mouse sensor measures the displacement (dx, dy) during a single timestep in its own coordinate system Σ_{Tamiya}. The state transition function $g(\mathbf{x}, \mathbf{u})$ provides, however, the differential dispacement in the world frame Σ_{world}. The transformation matrix H, which converts the state parameter into the measurement domain, is therefore a two-dimensional rotation matrix in homogenous form:

$$\hat{\mathbf{z}}_t = H\Delta\hat{\mathbf{x}}_t = \begin{pmatrix} cos(\theta_{t-1}) & -sin(\theta_{t-1}) & 0 \\ sin(\theta_{t-1}) & cos(\theta_{t-1}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_t \cdot dt \cdot cos(\theta_{t-1}) \\ v_t \cdot dt \cdot sin(\theta_{t-1}) \\ \frac{v_t \cdot tan(\phi_t)}{L} \cdot dt \end{pmatrix} \tag{2.14}$$

Correction step

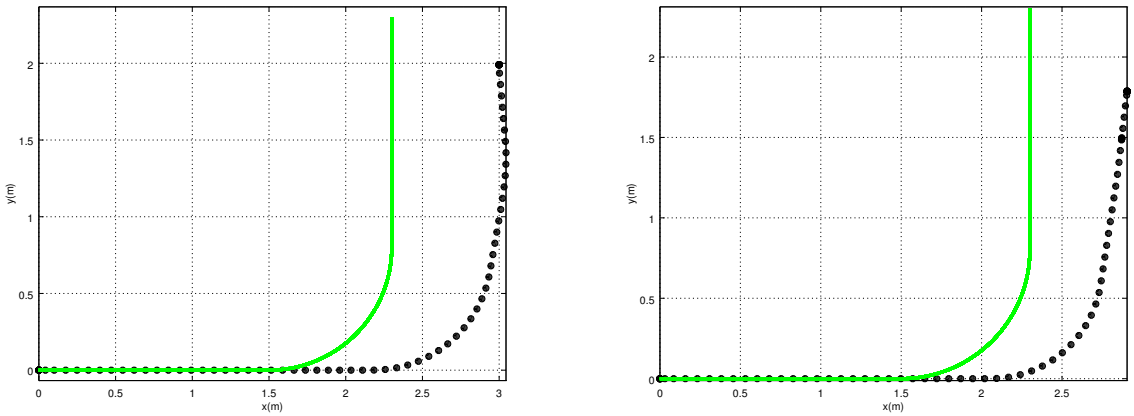
The correction step is performed, using equations 2.6 and 2.7.

The Extended Kalman Filter was programmed on the main computer system, instead on the microcontroller. This way the maximum sensor refresh rate could be utilised, resulting in a smaller discretisation error. A detailed analysis of the filter performance is shown in the next chapter.

3. Results

3.1. Mechanics model

The mechanics model was tested on the following track: the vehicle was propagated along a straight path for 1.5m and than a 90° left turn with a constant radius of 0.8m was performed. The car was driven straight for another 1.5m, resulting in a total track length of 4.25m with an end state at (2.30m, 2.30m, 1.57 rad). The test was performed twice. Although it is not possible for the operator to follow exactly the chosen trajectory and some deviation from the path occured, the obtained data was reliable enough to assess the strengths and weeknesses of the mechanics model. Figure 3.1 shows the results of the experiment. The test course is coloured in green and the trajectory, computed with the mathematical model from equation 1.2, is marked with the black dots:



(a) Test 1: end state (2.99m, 1.99m, 1.58 rad)

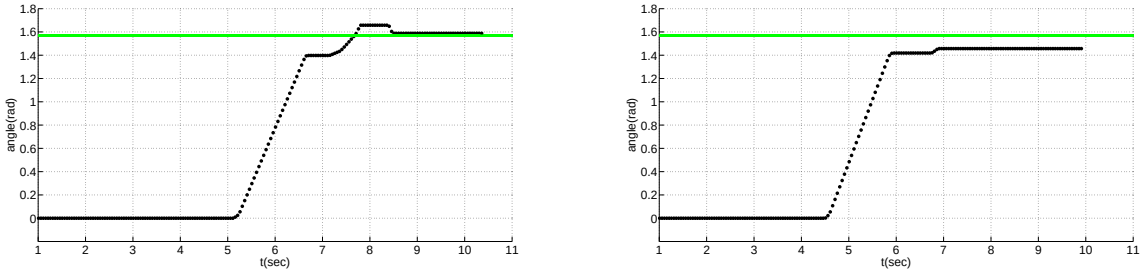
(b) Test 2: end state (2.90m, 1.79m, 1.46 rad)

Figure 3.1.: Mechanics model: position estimate; the test track with an end point at (2.30m, 2.30m, 1.57 rad) is coloured in green; the black dots represent the mechanics model

Both tests show a similar output. According to the mechanics model, the vehicle starts turning left at a further position along the x-axis, when compared to the executed path. This is explained by the fact, that the model does not take into account the vehicle acceleration. The resulting distance estimate will be greater than the actual one, because the model assumes, that the input velocity is achieved instantaneously. After that the simulation shows a change in direction, similar to the actual one. This means, that the estimated values for (v, ϕ), together with the system's equations, represent a satisfactory model of the vehicle dynamics. A significant difference in the executed trajectory and the proposed model arise at the end of the curve. As one can see from figure 3.1, in both cases the mechanics model predicts an earlier halt at y = 1.99m and y = 1.79m, compared to the actual end point with y = 2.30m. This behaviour is not surprising and is due to the fact, that near the end of the test the vehicle was moving without any velocity command as a result

of its momentum. The system calculates its path by using the input velocity and steering angle, therefore this type of motion will always reamain unaccounted for by this model.

Figure 3.2 shows the estimated angle θ as a function of the elapsed time. The deviation from the strictly linear progression is result of the operator's actions to steer the robot to the chosen end point. The green line marks the desired end value θ_{end} of 1.57 rad:



(a) Test 1: $\theta_{end} = 1.58$ rad

(b) Test 2: $\theta_{end} = 1.46$ rad

Figure 3.2.: Mechanics model: angle estimate; the green line represents the desired θ_{end} of 1.57 rad at the end of the test; the black dots show the angle from the mechanics model

Conclusion:

As the results show, the mechanics model is capable of producing an acceptable postition estimate. Its precision, however, is determined by a lot of factors. Some of them, such as the mapping of the pulse duration to velocity and steering angle, could be improved by extensive testing, but the most influential result from additional forces, acting on the vehicle. Therefore the mechanics model does not suffice to produce a reliable position tracking on its own, separate solution should be used in combination with it to compensate for the following effects:

- **measurement error:** the process of converting the pulse length to velocity and steering angle is associated with certain noise. It is possible to improve the model by further experiments, but this would not improve the overall performance drastically, because of the other uncertainties involved.
- **linkage sway:** the linkage of the steering has a certain degree of freedom. Despite constant input velocity and steering angle, the loose connections between the servo and the wheels could result in different steering angles at fixed pulse durations.
- **acceleration error:** the input control values do not account for the vehicle acceleration. The resulting error is especially evident, when there is a significant change in the velocity over a short period of time (e.g. accelerating from a still position)
- **wheel slip/skating:** the model relies on a good traction between the car's tyres and the underlying surface. Any effect, which is not result of the input control, but from the environment will not be perceived and accounted for.
- **vehicle inertia:** the vehicle inertia is not included in the mechanics model. This can lead to inaccuracies, when the input velocity decreases to zero, but the car has gained momentum. Additionaly, any other force, acting on the vehicle will introduce a relatively high deviation.

3.2. Sensor data

A direct estimation of the vehicle's state could be accomplished, relying only on the data from the sensors. This is possible by using the angle from the IMU to project the values (dx, dy) provided by the laser sensor from the local vehicle frame into the world frame. The sensors were tested on three different test courses, in order to determine their accuracy and overall performance. Each test was repeated 20 times to obtain enough information for statistical analysis.

In order to give a qualitative as well as a quantitative assessment of the sensor data, it is necessary to define parameters, which describe the accuracy of the performed measurements. First, a metric for the mean error Δd will be introduced, which shows the deviation of the measured mean values (μ_x, μ_y) from the desired endstate (x, y):

$$\Delta d = \sqrt{(x - \mu_x)^2 + (y - \mu_y)^2} \tag{3.1}$$

This value alone does not provide enough information; the quotient of Δd and the total travelled distance s is required to evaluate the accuracy of the sensor with respect to the traversed path:

$$\varepsilon_\mu = \frac{\Delta d}{s} \tag{3.2}$$

A qualitative interpretation of the obtained measurement variance will be provided by the ratio of the standard deviation σ and the travelled distance s:

$$\varepsilon_\sigma = \frac{\sigma}{s} \tag{3.3}$$

Both the IMU and the mouse sensor have a lot of configuration registers, which alter their behaviour and therefore their precision. All of the following tests were performed using the same configuration, described below. It is optimised for the fastest data acquisition and uses the default settings for both sensors:

- accelerometer sensitivity: ±2g
- gyroscope sensitivity: ±2000°/s
- sample rate: 100Hz
- laser sensor resolution: 1620 cpi

When using the MPU-6050's Digital Motion Processor neither the accelerometer's, nor the gyroscope's full-scale range can be changed from those default values. The sample rate can be set from 20Hz to 100Hz. The ATMega328P controller serves only for data acquisition, so it is possible to utilise the highest update frequency. Increasing the resolution of the optical sensor did not improve its performance, but did impose a lower maximum speed limit. This is why the parameter is left to its default value.

3.2.1. Test course 1: straight line

ADNS-9500

A direct test to observe the distance measurement precision of the mouse sensor is to propagate it along a straight line with a fixed length and record the measured values for x and y. This experiment was performed for two distances with lengths of 2m and 5m. The exact results of the experiment are listed in the appendix, table A.1.

A visual representation of the data is obtained by plotting the measurement values on the xy-axis, as shown of figure 3.3. The green markers on the graph indicate the end coordinates, to which the vehicle was brought to a halt, namely (2.00, 0.00) and (5.00, 0.00). The black dots represent the position in (x, y), measured by the sensor. The data is also used to construct a covariance ellipse, with axes coloured in red and center, positioned at the mean value.

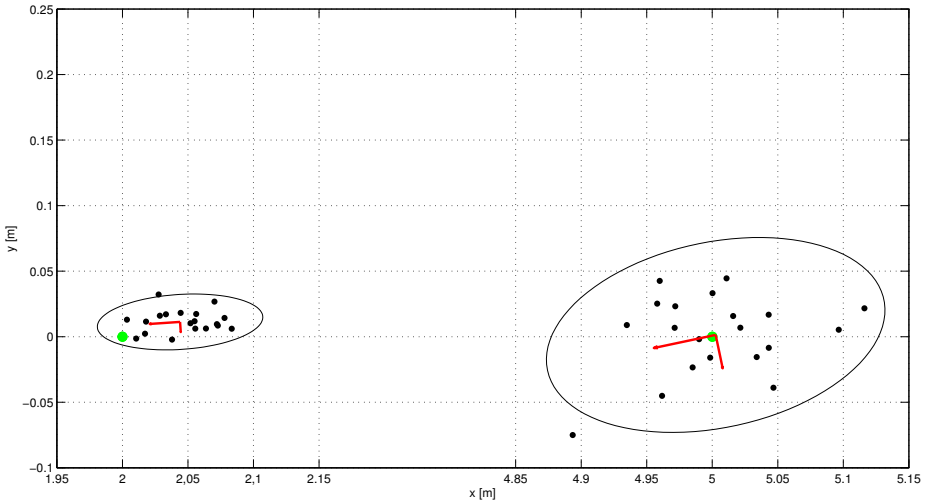


Figure 3.3.: ADNS-9500: 2m and 5m straight line test results: the measurement are shown in black; the green circle marks the end point; the red lines show the ellipse axes

2m line:

Table 3.1.: ADNS-9500: 2m line test results

μ_x	μ_y	σ_x	σ_y	Δd	ε_μ	ε_{σ_x}	ε_{σ_y}
2.04m	0.01m	0.025m	0.009m	0.041m	2%	1.2%	0.4%

The mean values and the standard deviations, together with the error parameters, defined in section 3.2, are summed in table 3.1. The measurements show mean values (μ_x, μ_y) of (2.04m, 0.01m) and standard deviations (σ_x, σ_y) of (0.025m, 0.009m). These values correspond to a mean error $\Delta d = 0.041$m, $\varepsilon_\mu = 2\%$ and $\varepsilon_\sigma = (1.2\%, 0.4\%)$ for x and y respectevely.

5m line:

Table 3.2.: ADNS-9500: 5m line test results

μ_x	μ_y	σ_x	σ_y	Δd	ε_μ	ε_{σ_x}	ε_{σ_y}
5.002m	0.001m	0.052m	0.03m	0m	0%	1%	0.6%

Table 3.2 shows the results for the 5m test line. As the track length more than doubles, the measurements remain still relative precise, registering mean values of (5.002m, 0.001m) and standard deviations of (0.052m, 0.03m). This point distribution leads to $\Delta d \approx 0m$, $\varepsilon_\mu \approx 0\%$ and ε_σ of (1%, 0.6%).

These results indicate high consistency of the performed measurements. For both test lengths the mean estimate is very close to the actual coordinates and the standard deviation increases proportionally to the travelled distance.

ADNS-9500 and MPU-6050

The straight line test was performed again, this time by combining the ADNS-9500 displacement data with the current orientation, provided by the IMU. Measurements on an additional track with length 10m was also carried out. The measured values are listed in table A.2 in the appendix. Figure 3.4 shows the obtained results; each data set uses the same axis scale, so that the three point distributions can be visually compared:

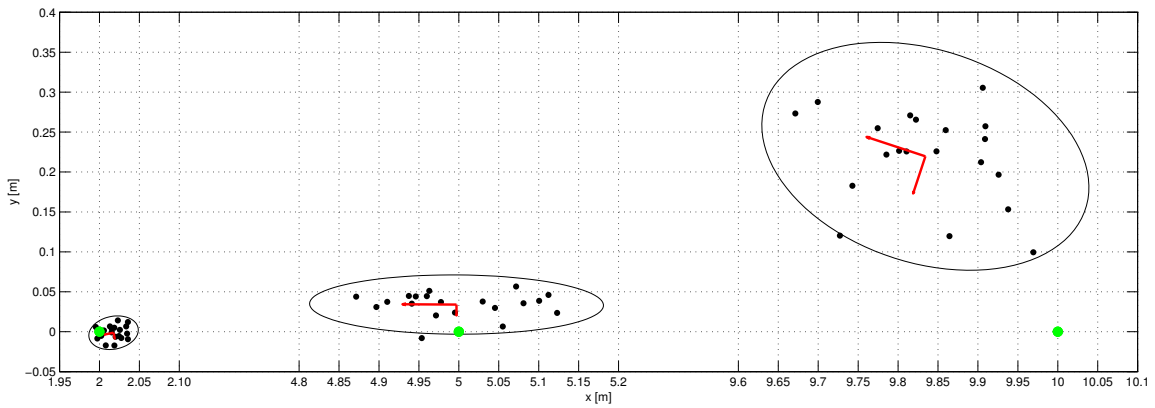


Figure 3.4.: ADNS-9500 and MPU-6050: 2m, 5m and 10m straight line test results: the measurements are shown in black; the green circle marks the end point; the red lines show the ellipse axes

Tables 3.3, 3.4, and 3.5 summarise the measured mean values, standard deviations and error factors.

2m line:

Table 3.3.: ADNS-9500 and MPU-6050: 2m line test results

μ_x	μ_y	σ_x	σ_y	Δd	ε_μ	ε_{σ_x}	ε_{σ_y}
2.017m	-0.001m	0.012m	0.0086m	0.017m	0.85%	0.6%	0.43%

The acquired data shows mean values close to the real end point with a standard deviation of less than 2cm for both axes. This high consistency in the measurements results in error factors $\varepsilon_\mu, \varepsilon_{\sigma_x}, \varepsilon_{\sigma_y}$ below 1%.

5m line:

Table 3.4.: ADNS-9500 and MPU-6050: 5m line test results

μ_x	μ_y	σ_x	σ_y	Δd	ε_μ	ε_{σ_x}	ε_{σ_y}
4.997m	0.034m	0.075m	0.015m	0.034m	0.68%	1.5%	0.3%

As the results indicate, the variance in the x-values increases, but the one in the y-values remains low. As the track length increases, the additive character of the distances measurement causes the uncertainty to grow. Since there is significant less movement in the y-direction, the variance along this axis does not increase much. The y-axis mean value, however, is not close to zero: this can be explained by the fact, that the angle from the MPU-6050 plays a crucial role in the measurements. Even if the laser sensor measurements were 100% accurate, any initial offset of the IMU from 0° will result in a small value, calculated for the y-axis. Based on the acquired data, a 3.4cm y-offset for a 5m straight line corresponds to a 0.389° initial angle, which is practically impossible to compensate, when aligning the vehicle to the 5m test line. The error factors, calculated from this data, remain below 2%, meaning that even though the variance increases with the length of the track, it grows proportionally to the distance.

10m line:

Table 3.5.: ADNS-9500 and MPU-6050: 10m line test results

μ_x	μ_y	σ_x	σ_y	Δd	ε_μ	ε_{σ_x}	ε_{σ_y}
9.834m	0.219m	0.083m	0.058m	0.275m	2.75%	0.83%	0.58%

The obtained data has a mean value at (9.834m, 0.219m), which for the length of the test course is equivalent to a ε_μ of 2.75%. The standard deviations grow larger with the increase of the travelled distance, but still represent less than 1% of the total path length, as shown by ε_{σ_x} and ε_{σ_y}. A 22cm mean offset from the 0 value for the y-axis is equivalent to a 1.2581° initial vehicle offset from the straight 10m line, which again is difficult to determine, while positioning the robot on the test track.

3.2.2. Test course 2: circular track

The second test track represents an arc with a radius of 0.8m. The purpose of this track is to determine how accurate the measured displacement values from the ADNS-9500 are projected into the world frame, using the angle from the MPU-6050. Three series of tests were performed to evaluate the performance of the sensor combination:

- 180° left turn with path length of 2.5m
- 360° left turn with path length of 5m
- 720° left turn with path length of 10m

The test results are summed in table A.3 in the appendix. Figure 3.5 illustrates the test track with the point distributions for each test.

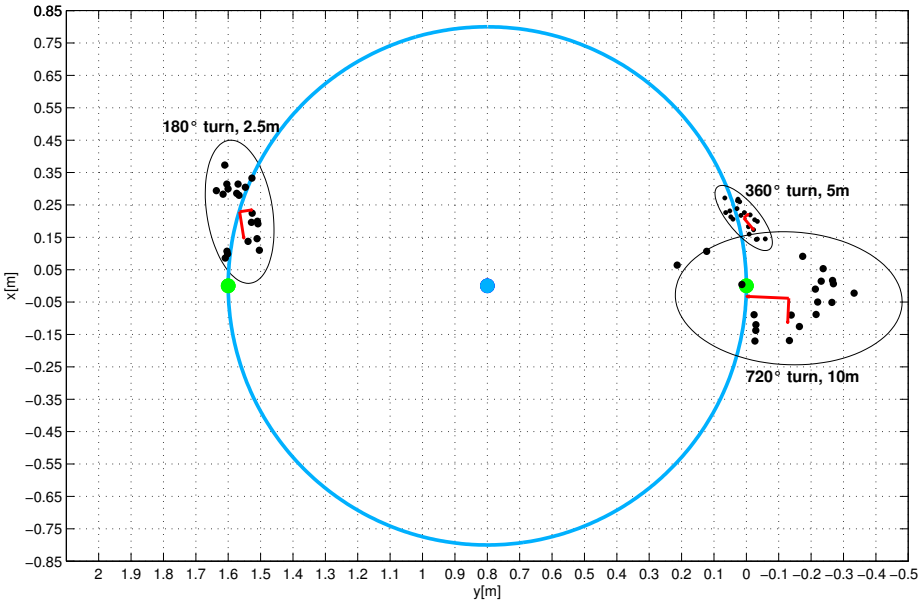


Figure 3.5.: ADNS-9500 and MPU-6050: 180°, 360°, 720° arc test results: the measurements are shown in black; the green circles mark the end points at 0°(right) and 180°(left); the red lines show the ellipse axes; the vehicle movement is counter clockwise

For each of the three test lengths the vehicle started from (0.00m, 0.00m), represented by the green dot on the right on figure 3.5. It followed the path, traced by the blue line in counter clockwise direction, and was brought to halt at (0.00m, 1.60m) for the 180° turn. The end point is marked with the green dot on the left side. For the 360° turn a full circle was performed and for the 720° test the vehicle completed two rotations with a total travelled distance of 10m, ending its movement at the start point. The black dots with their corresponding ellipses show the measured position values and the red lines represent the ellipse axes. A detailed analysis of the data is obtained by reviewing the mean values μ_x, μ_y, the standard deviations σ_x, σ_y and the error factors Δd, ε_μ, ε_{σ_x}, and ε_{σ_y} for each test series. These parameters are given in tables 3.6, 3.7, and 3.8.

180° turn, 2.5m:

Table 3.6.: ADNS-9500 and MPU-6050: 180° turn test results

μ_x	μ_y	σ_x	σ_y	Δd	ε_μ	ε_{σ_x}	ε_{σ_y}
0.229m	1.564m	0.043m	0.09m	0.232m	9.2%	1.7%	3.5%

Analysing the results of the first experiment of the series, leads to the observation, that the standard deviation of the measurements is larger, compared to the results from a straight line of a similar length (see section 3.2.1). As each dispacement value from the laser sensor is rotated, based on the current angle, measured by the IMU, the uncertainty in the angle measurement has its effect on the overall sensor performance.

360° turn, 5m:

Table 3.7.: ADNS-9500 and MPU-6050: 360° turn test results

μ_x	μ_y	σ_x	σ_y	Δd	ε_μ	ε_{σ_x}	ε_{σ_y}
0.209m	0.008m	0.041m	0.036m	0.209m	4.1%	0.8%	0.7%

As figure 3.5 indicates, the measurements are distributed in the vicinity of the end point. Although the standard deviation of the data is smaller, compared to the 180° test, the measurement errors lead to a trajectory, which is resembles a spiral, not a circle, so that the point distribution alone is not enough for a qualitative assessment.

720° turn, 10m:

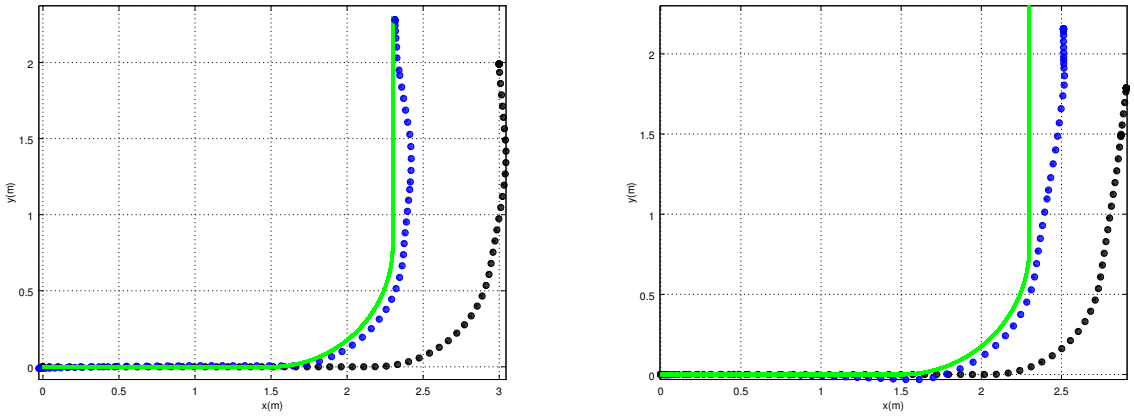
Table 3.8.: ADNS-9500 and MPU-6050: 720° turn test results

μ_x	μ_y	σ_x	σ_y	Δd	ε_μ	ε_{σ_x}	ε_{σ_y}
-0.038m	-0.131m	0.084m	0.143m	0.136m	1.3%	0.8%	1.4%

The more widely spaced out samples for the 720° test course in figure 3.5 indicate data with a larger variance. Similar to the previous tests, the standard deviation increases with the distance. Nevertheless, after two complete circles, the position provided by the sensor combination registers a mean value, located 13.6cm from the actual end point. The small error factors ε_μ, ε_{σ_x} and ε_{σ_y} of less than 1.5% confirm the precision of the selected pair of sensors.

3.2.3. Test course 3: curved path

The last sensor test uses the same track, on which the mechanics model was tested. This way a direct comparison of the two approaches could be accomplished. The sensor data was recorded during the the experiments in section 3.1. Figure 3.6 shows the path computed by the mechanics model in black and the one from the sensor combination in blue. The course, which the operator followed is coloured in green.



(a) Test 1: end state mechanics (2.99m, 1.99m, 1.58 rad); end state sensors (2.31m, 2.28m, 1.61 rad)

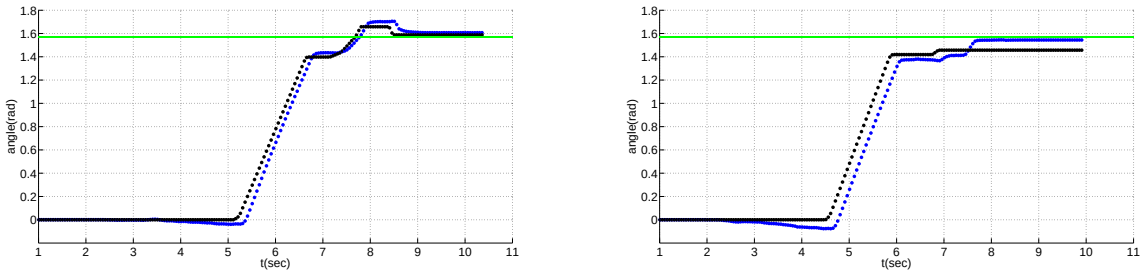
(b) Test 2: end state mechanics (2.90m, 1.79m, 1.46 rad); end state sensors (2.51m, 2.15m, 1.54 rad)

Figure 3.6.: Mechanics model and sensor data: position estimate; the test track with an end point at (2.30m, 2.30m, 1.57 rad) is coloured in green; the black dots represent the mechanics model; the trajectory from the sensors is marked in blue

The output of the sensor combination shows a curve, much similar to the actual test track. In both cases the trajectory closely resembles the path, which it is supposed to follow. Under conditions, which include a flat, non-reflecting surface and a medium vehicle speed, the sensor measurements indicate high reliability. The end coordinates, calculated by the sensors in the first test, measure a difference of 2cm from the desired end state and 2° from the desired orientation. In the second experiment this difference is larger - 26cm from the actual end point, but the trajectory is consistent with the data from the mechanics model in terms of direction of travel and final position, meaning that the operator strayed from the planned path.

A significant advantage of measuring the travelled distance with the mouse sensor is the fact, that the dispacement of the vehicle relative to the ground is measured with high precision. This measurement method also covers the effect of other forces, acting on the robot, so any movement will be perceived as such, regardless of the input controls. As a result, according to the sensor combination, the vehicle starts turning at a point, very close to the actual one. Furthermore, in the absence of an input velocity, movement, caused by the gained momentum is still detected, as the laser sensor continues scanning the surface.

Figure 3.7 shows a comparison between the angle estimates from the mechanics model and the IMU over time:



(a) θ_{end}=1.58 rad (mechanics); θ_{end}=1.61 rad (IMU)

(b) θ_{end}=1.46 rad (mechanics); θ_{end}=1.54 rad (IMU)

Figure 3.7.: Mechanics model and sensor data: angle estimate; the green line represents the desired θ_{end} of 1.57 rad at the end of the test; the black dots show the angle from the mechanics model; the IMU angle estimate is coloured in blue

The IMU data exhibits a similar output to the mechanics model. Although it shows a certain angle offset, as the results from section 3.2.1 suggest, the sensor is capable of detecting rotations, which are not caused by the control commands. The sensor combination does have its limitations though:

- **gyro drift:** the gyroscope of the IMU drifts over time. The signal processor can correct the drift in the (x, y) direction by using the gravity, measured by the accelerometer, but no such correction is possible for the yaw angle.
- **lens contamination:** the laser sensor requires a relatively flat, clean surface for correct operation. If there is dirt on the lens, or any other obstruction, the displacement value provided by the sensor will be incorrect.
- **change of height:** the mouse sensor will also deliver inaccurate data, if its distance from the ground changes. A bump or a dent on the track will result in a series of wrong measurements. Correct operation will be established, when the distance to the ground is restored to normal.

Conclusion:

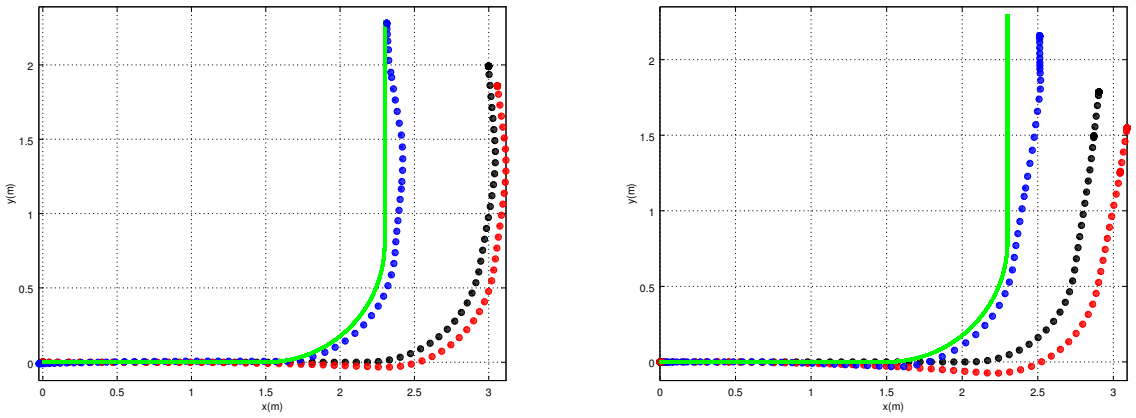
The sensor data could however minimise some of the errors, which the mechanics model cannot. First of all, it is independent from the used platform: the motion processor of the IMU delivers data about the sensor's orientation, regardless of how rotation was introduced to the system. The mouse sensor measures the displacement relative to the ground beneath it, so as long as the vehicle is situated on the ground, every movement will be detected. The independence on the underlying mechanics, whether it is Ackermann steering, differential drive or some other platform, allows the combination of sensors to detect the changes of the robot's state, which are not a direct effect of the input control. Therefore the car's momentum and wheel slipping/skating could also be accounted for. On the other hand, the mechanics model could compensate the gyro drift over time, since in the absence of movement no change in orientation is expected. Moreover the laser sensor precision is dependent on a lot of parameter, such as flat surface, constant distance from the ground, etc., which could change in the course of operation, while the input control commands remain impartial to the terrain. Therefore by combining the data from both models in an adequate manner, both approaches could be used in a way to compensates the disadvantages of each other. This was achieved by using an Extended Kalman Filter as a data fusion algorithm.

3.3. Extended Kalman Filter

An improved estimate of the vehicle state could be obtained by combining the the mechanics model with the sensor data. The algorithm used for the data fusion is the Extended Kalman Filter, described in section 2.2.2. Its role is to provide a weighted average of the robot state, based on the information from the control input and the sensor measurements in a way to decrease the uncertainty in the position estimate.

Filter output with an angle correction:

The first test examines the performance of the system, when the angle from the mechanics is corrected by the value from the IMU. This behaviour will be modelled in the filter algorithm by increasing the control input noise and decreasing the measurement noise. This way the data from the IMU will have a higher influence on the angle estimate. Figure 3.8 shows a comparison of the mechanics model (in black), the sensor combination (marked in blue), and the filter output (coloured in red):



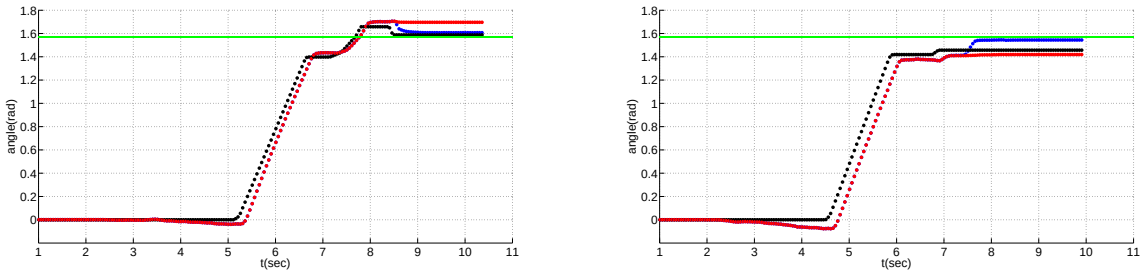
(a) Test 1: filter state (3.06m, 1.86m, 1.69 rad)

(b) Test 2: filter state (3.09m, 1.55m, 1.42 rad)

Figure 3.8.: Angle correction with EKF: position estimate; the test track with an end point at (2.30m, 2.30m, 1.57 rad) is coloured in green; the black dots represent the mechanics model; the trajectory from the sensors is marked in blue; the filter output is shown in red

As illustrated by figure 3.8, the filter generates a trajectory, similar to the one from the mechanics model, but with a curvature, which resembles the output from the sensor combination. If the uncertainty in the mechanics model is decreased or the noise in the IMU measurement is increased, the filter output will shift towards the state, predicted by the mechanics (the black trajectory). This ability of the filter algorithm to change the preffered data source could be utilised to compensate the drift of the IMU gyroscope, when there is no input velocity command.

More detailed information of the filter performance is obtained by comparing the change of the angle over time, computed by the filter, to the angles, provided by the mechanics and the IMU. Figure 3.9 shows the angles from the mechanics, the IMU and the filter in black, blue and red respectevely:



(a) Test 1: filter angle: θ_{end}=1.69 rad

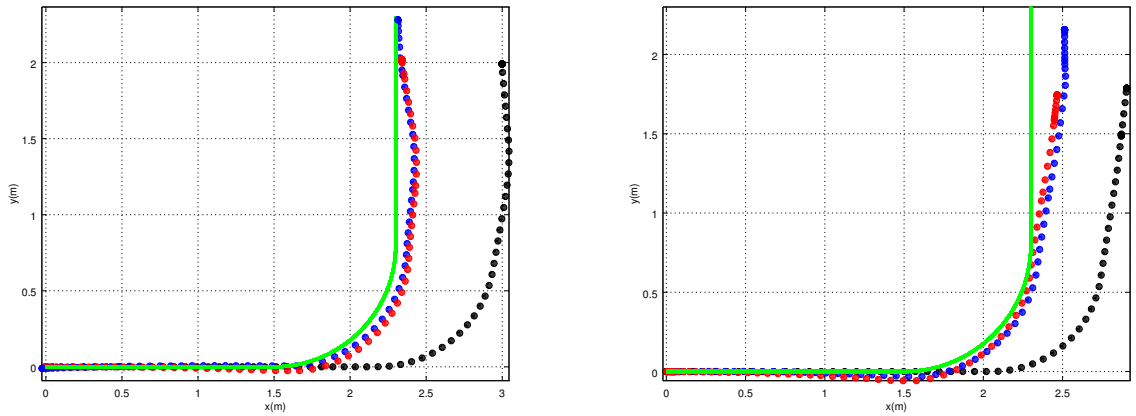
(b) Test 2: filter angle: θ_{end}=1.42 rad

Figure 3.9.: Angle correction with EKF: angle estimate; the green line represents the desired θ_{end} of 1.57 rad at the end of the test; the black dots show the angle from the mechanics model; the IMU angle estimate is coloured in blue; the filtered result is marked in red

In both tests the filter output initially follows the angle, given by the MPU-6050, as indicated by the overlapping of the plots. Near the end of the simulations, despite the change in the angle, detected by the IMU, the value from the filter remains constant. This observation is explained by the following fact: the filter output depends on the noise associated with each model. In the last couple of seconds the vehicle does rotate a little, as measured by the sensor. But since there is no input velocity (the robot is using its current momentum to steer), the noise from the mechanics model is reduced to a small value. This behaviour is expected, because without an input velocity the mechanics model does not exhibit any noise. As a result, the last sensor updates are discarded - the confidence in the mechanics is greater, due to the fact that no movement is registered.

Filter output with position and angle correction:

Figure 3.10 shows the filter performance with an additional correction in the measured distance:



(a) Test 1: filter state (2.34m, 2.02m, 1.69 rad)

(b) Test 2: filter state (2.47m, 1.74m, 1.42 rad)

Figure 3.10.: Position and angle correction with EKF: position estimate; the test track with an end point at (2.30m, 2.30m, 1.57 rad) is shown in green; the mechanics model is coloured in black; the sensors' estimate is marked in blue; the filter output is shown in red

The results of the laser sensor measurements in section 3.2.1 justify the decision to assign small uncertainty to the mouse sensor. By increasing the confidence in the measurements, a filter behaviour, similar to the angle correction, is observed: the filter output follows the sensor data right up to the last couple of seconds. At the end, the zero input velocity reduces the noise in the mechanics model and forces the filter to ignore the position updates from the laser sensor.

Conclusion:

As indicated by the results, the filtered values do not provide an immediate solution to the position estimation problem. There are situations, where the dispacement from the mechanics $v \cdot dt$ and the dispacement from the laser sensor (dx, dy) contradict, which demand additional changes to the proposed model:

- $v \neq 0$, $(dx, dy) \neq 0$: this is the case for normal operation, where the filter output will depend on the predefined uncertainties.
- $v = 0$, $(dx, dy) = 0$: both models do not detect movement; the robot is standing still.
- $v = 0$, $(dx, dy) \neq 0$: this situation occurs, when the laser detects movement with zero input velocity. This happens, when the vehicle is rolling due to its inertia, or the wheels have lost traction and the car is sliding. In this case the filter should adopt the sensor measurements, although the noise in the mechanics is small. This could be accomplished by including an additional term in the predicted covariance, such as the velocity is zero, the noise gets large enough, so that the mechanics data is practically ignored.
- $v \neq 0$, $(dx, dy) = 0$: if the output torque on the wheels is too high, the vehicle will skid. The laser sensor will not detect a change in position, so that the measurement should be preferred over the mechanics. However, this difference in the velocity could arise if the laser sensor lens is contaminated by dirt, or bumps on the track change its distance to the ground. In these cases the mechanics will be a better estimate of the vehicle movement and the filter should minimise the influence of the measurements. The model in its current setup cannot resolve this conflict situation.

In order to take advantage of the EKF algorithm, several system changes have to be made. First, the state should be extended to include the current velocity, so that the filter corrects the velocity, rather than the dispacement directly. This means, that a separate velocity measurement is required. This could be accomplished by integrating the acceleration from the IMU: although it is relatively noisy, it could serve the purpose of determining, whether the robot is moving or not. This approach, however, was not implemented in this project: the measured acceleration showed after calibration an angle dependent offet, which means that even for a stationary robot the acceleration and therefore the velocity will change according to the current orientation.

A position tracking system will become increasingly complex, when trying to compensate for all of the possible scenarios, environments and additional factors, which determine the accuracy of the position estimation. Exhaustive research and thorough testing are required for the design of a complete functional prototype, which is not feasible within the given time span. As the test results indicate, an EKF algorithm could be practically used to combine adequately data from the input commands for an Ackermann steering vehicle and the measurements from an IMU and a laser mouse sensor to provide a weighted estimate of the robot's state. Further improvement in this direction should be the topic of another work.

4. Summary

The position tracking system developed in this project combines the input velocity and steering commands with the data from an IMU and a mouse sensor in an Extended Kalman Filter to estimate the coordinates of a mobile robot. Several tests of the system were performed to determine the system advantages and limitations, as well as to propose alternative methods for compensating some of the design flaws.

The experiment results indicate precise sensor measurements for distances up to 10m with mean error of around 3% of the total travelled distance. The obtained data exhibits also small variance, with a ratio of the standard deviation to the total traversed path ranging from 0.3% to 3.5%. The integration of the Extended Kalman Filter provides a mechanism to select between different position information sources based on their noise. The filter can be extended to use data from additional sensors for optimised state estimation. As none of the original components of the test vehicle were modified or removed, the system can be potentially implemented on different robot platforms due to its simple hardware design. Moreover, as the system performs as a regular ROS node, it can be easily extended to participate in more complex robot implementations, such as autonomous navigation in known and unknown environments.

The system limitations are mostly connected to the sensor imperfections and the incomplete filter model. The mouse sensor should be mounted several millimeters above a flat and non-reflecting surface and requires constant distance to the ground for proper measurements. The IMU gyroscope exhibits a certain drift over time, resulting in an inaccurate angle estimate. The implementation of the Extended Kalman Filter needs to be revised to account for different movement scenarios.

The system could be further developed and improved to compensate some of the mentioned drawbacks. An additional optical lens with specific focal length could be used to increase the distance between the laser sensor and the ground. Introducing a separate sensor for speed measurement could provide further information about the motion of the vehicle and can be used to resolve conflict situations, due to contradicting data from the model and the laser sensor, as described at the end of section 3.3. The state model used in the Extended Kalman Filter could be extended to include the velocity of the vehicle in the correction step, instead of correcting the robot displacement directly. Detailed examination of the filter performance in different situations would provide feedback to customise the algorithm for optimal use with the selected mechanics model and sensors.

Despite the limitations, the system demonstrated high reliability in tracking the position of the remote control vehicle, and it is suitable for implementation in small vehicles for indoor operation since it can be mounted without modification of the chassis.

A. Sensor measurements

Table A.1.: ADNS-9500: straight line results

ADNS-9500				
	2m		5m	
	x	y	x	y
1	2,0034	0,013	5,0431	-0,0085
2	2,018	0,0115	5,0159	0,0158
3	2,0444	0,0182	5,0002	0,0332
4	2,0519	0,0103	5,0109	0,0445
5	2,0378	-0,0022	5,0214	0,0069
6	2,055	0,0118	4,99	-0,0019
7	1,9999	0,0014	4,9599	0,0426
8	2,0276	0,0322	4,9984	-0,016
9	2,0285	0,016	4,9348	0,0089
10	2,0721	0,0096	5,1161	0,0217
11	2,0779	0,0143	4,958	0,0252
12	2,0702	0,0268	4,8935	-0,075
13	2,0728	0,0085	5,043	0,0168
14	2,0563	0,0173	4,985	-0,0234
15	2,0105	-0,0014	5,0339	-0,0155
16	2,0332	0,0171	5,0467	-0,0389
17	2,0556	0,0062	5,0965	0,0053
18	2,0172	0,0023	4,9617	-0,0451
19	2,0637	0,00628	4,9713	0,0068
20	2,0834	0,0061	4,9718	0,0232
μ	2,04397	0,011264	5,002605	0,00133
σ^2	0,00066518	0,00007585	0,00278425	0,00092268

Table A.2.: ADNS-9500 and MPU-6050: straight line results

MPU-6050 with DMP and ADNS-9500						
	2m		5m		10m	
	x	y	x	y	x	y
1	2,0231	0,0142	4,8968	0,031	9,8013	0,2265
2	2,0022	-0,0051	4,9461	0,0442	9,8481	0,2258
3	2,0256	0,0024	4,9776	0,0372	9,8596	0,2524
4	2,0186	0,005	5,1005	0,0387	9,8152	0,2709
5	1,9955	0,0059	4,96	0,0446	9,7745	0,2548
6	2,0203	-0,0063	4,963	0,0511	9,7854	0,2218
7	2,0335	0,0067	5,055	0,0065	9,7272	0,1203
8	2,0131	0,0066	4,9103	0,0374	9,9094	0,2573
9	2,0356	-0,0094	5,1231	0,0236	9,9693	0,0995
10	2,0274	-0,0078	4,9951	0,024	9,9378	0,1532
11	2,0187	-0,0172	5,0452	0,0298	9,9089	0,2413
12	1,9973	-0,0085	5,1123	0,04612	9,8644	0,1197
13	2,0156	0,0036	4,9374	0,0448	9,6994	0,2877
14	2,0153	-0,0009	5,0717	0,0566	9,6714	0,2733
15	2,0035	-0,0026	4,8714	0,0439	9,9259	0,1966
16	2,0356	0,0121	4,9411	0,0352	9,8224	0,2655
17	2,0346	-0,0024	5,0299	0,0378	9,7428	0,1828
18	2,0245	-0,0054	4,9536	-0,0079	9,9061	0,3054
19	2,0079	-0,0171	5,081	0,0357	9,8105	0,2257
20	2,0056	0,0016	4,9714	0,0204	9,9039	0,2122
μ	2,017675	-0,00123	4,997125	0,034036	9,834175	0,219635
σ^2	0,00016006	0,00007396	0,00565037	0,00023028	0,00699867	0,00339832

Table A.3.: ADNS-9500 and MPU-6050: circular path results

MPU-6050 with DMP and ADNS-9500									
	180°			360°			720°		
	x	y	θ	x	y	θ	x	y	θ
1	0,0988	1,602	180,405	0,1825	-0,0065	-0,672	-0,1703	-0,0264	1,644
2	0,1099	1,5037	178,921	0,2041	-0,0259	0,576	-0,089	-0,024	-0,699
3	0,1457	1,5111	173,574	0,26	0,0207	-1,21	-0,1255	-0,1639	0,441
4	0,1964	1,5282	180,117	0,263	0,0272	0,664	-0,0884	-0,2154	1,245
5	0,1374	1,5388	179,516	0,213	0,048	-1,287	0,0043	0,0137	1,007
6	0,0858	1,6092	180,194	0,2193	-0,0119	-1,014	0,0532	-0,2371	0,007
7	0,3145	1,6042	179,25	0,2175	0,0169	-1,063	0,0168	-0,2663	1,294
8	0,283	1,6161	178,985	0,2316	0,0512	-0,79	0,0642	0,2137	0,658
9	0,3145	1,5698	179,083	0,1734	-0,0225	0,098	-0,0499	-0,2201	1,728
10	0,107	1,6038	178,6	0,1599	-0,0088	-0,609	0,0912	-0,1741	0,923
11	0,1918	1,5084	178,837	0,2265	0,064	-1,147	-0,0223	-0,3326	0,378
12	0,294	1,6368	179,642	0,2388	0,0287	-0,307	0,0066	-0,2689	1,945
13	0,2857	1,5737	178,264	0,1435	-0,0305	-0,979	0,0145	-0,2313	1,245
14	0,2244	1,5261	179,069	0,2677	0,0258	0,077	-0,1198	-0,0296	1,462
15	0,3729	1,6105	178,907	0,2264	0,0064	-0,595	-0,1688	-0,133	-0,378
16	0,3047	1,5471	177,683	0,2713	0,0666	-0,21	-0,0101	-0,2132	2,098
17	0,2798	1,5667	177,774	0,1991	-0,034	-0,468	-0,0903	-0,1386	0,979
18	0,333	1,5269	178,138	0,2061	0,0413	-0,923	-0,0508	-0,2644	1,273
19	0,2997	1,6004	177,732	0,1452	-0,0584	-1,182	-0,1375	-0,0292	1,224
20	0,1995	1,5099	180,138	0,1448	-0,034	-1,224	0,1067	0,1227	1,728
μ	0,228925	1,56467	178,74145	0,209685	0,008215	-0,61325	-0,03826	-0,1309	1,0101
σ^2	0,00815809	0,00188328	2,16480479	0,00168579	0,00130991	0,35392472	0,00705241	0,02047651	0,56369525

5. Bibliography

[1] Ahmed El-Rabbany, *Introduction to GPS The Global Positioning System*, 2002.

[2] J. Borenstein, H.R. Everett, L. Feng, D. Wehe, *Mobile Robot Positioning: Sensors and Techniques*, University of Michigan, 1996.

[3] Bong-Su Cho, Woo-sung Moon, Woo-Jin Seo and Kwang-Ryul Baek, *A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding*, Pusan National University, 2011.

[4] Vikas Choudhary, Krzysztof Iniewski, *MEMS: Fundamental Technology and Applications*, 2013.

[5] Zheyuan Lu, Zhencheng Hu, Keiichi Uchimura, *SLAM estimation in dynamic outdoor environments*, 2010.

[6] Caius Suliman, Cristina Cruceru, Florin Moldoveanu, *Mobile Robot Position Estimation Using the Kalman Filter*, Transilvania University of Brasov, 2009.

[7] Pierre Lamon, *3D-Position Tracking and Control for All-Terrain Robots*, 2008

[8] Dennis Moy, *Optical mouse technology: Here to stay, still evolving*, 2011.

[9] Espressif Systems, *Espressif smart connectivity: platform ESP8266*, Product specifications, 2013

[10] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng *ROS: an open-source Robot Operating System*, 2009.

[11] Greg Welch, Gary Bishop, *An Introduction to the Kalman Filter*, University of North Carolina, 2004.

[12] Shu-Li Sun, Zi-Li Deng, *Multi-sensor optimal information fusion Kalman filter*, Heilongjiang University, 2004.

[13] Hisashi Tanizaki, *Nonlinear Filters: Estimation and Applications*, 2013.

[14] PixArt Imaging Inc., *ADNS-9500 Laser Gaming Sensor*, Product manual, 2013.

[15] InvenSense Inc., *MPU-6000 and MPU-6050 Product specification Revision 3.4*, 2013.

[16] InvenSense Inc., *MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.0*, 2012.

[17] Atmel Corporation, *ATMEL 8-bit mirocontroller with 4/8/16/32KBytes in-system programmable flash* Datasheet, 2014.

[18] Tamiya Inc., *TT-02 Radio control 4WD high performance racing car* Product manual, 2013.

[19] Ying Xu, *High speed non-holonomic mobile robot online trajectory optimization* John Hopkins University, 2014.

[20] Zachary Scheffer, *Optical speedometer*, University of Central Florida, 2007.