Notes on Robot Odometry

Gareth Cawood
Mechatronics Dep., Nelson Mandela Metropoliton University *

December 13, 2012

Abstract

In this report a pre-built omnidirectional robot is modified to allow for better odometry as well as control. The report investigates several possibilities for odometry, and then making use of available equipment adds a proximity sensor to the robot for accurate tracking. In addition to this the movement control code is replaced with PID controllers for both directions of movement. Slight adjustments are made to the visual interface with the robot and a Wi-Fi connection is re-established.

1 Introduction

This report covers the investigation of methods to obtain accurate odometry for the movement of a mobile robot developed by the Robotics Lab at Reutlingen University. The development of the robot to its current state comprised two separate bachelor's projects for mechatronics students.

The report gives an overview of the previous projects, it then looks at the current problems with the system and researches the accuracy of the system. From this different potential solutions to the problems are researched.

Finally the changes to the system are documented. This includes the reintroduction of a wireless connection to the robot from a computer. It also entails the testing of two separate proximity sensors, changes to the visual interface with the robot as well as the updating of the control methods used.

*Completed at Reutlingen University under supervision of Prof Dr-Ing Gruhler

1.1 Previous Work

A project to build a mobile robot capable of omni-directional motion in two planes was undertaken by Weiß (2009). The project was successful and the final product comprised of a Berghof CAN PLC controller, which drove two servo motors via individual Dematek servo motor drivers. These servo motors rotated a ball which was in contact with the ground which provided the drive.

While the device performed well, the drive method resulted in occasional slip. A visual interface allowed one to program in a point to move to, in X and Y coordinates. When the robot tried to move to that point it would not always reach it. It is also to be noted that a joystick plugged into the robot can be used to operate the platform.

A second project by Oßmann (2009) extended on this project. It built in wireless connections via both Wi-Fi and XBee Series 2 modules. It also included an accelerometer to attempt to determine location via monitoring inertial movements.

From the report programming over Wi-Fi was successful. Infrastructure changes at the university however meant that the Wi-Fi connection was no longer available. Furthermore the report claims that during testing the XBee unit as well as some other electronics received some damage which currently put them out of order, although no connection between the XBee and the controller was successful before this point.

The accelerometer was successfully implemented and data read. The report states that the sensor information was unfortunately not of sufficient quality to use for control. The sensor was prone to drifting and noise in the data meant it was not useful.

It is unclear at this moment whether data can still be obtained from the sensor. Monitoring the inputs on PLC no change is noted. It was suggested that a more expensive sensor could solve the current problems with the read values and result in better control. The project also gave the ability to program in nine points of movement which the robot moves towards in a linear fashion.

1.2 Problem Statement

The main problem that required a solution was determining how far the robot had actual travelled during a movement. It is quite easy to give the correct commands to the motors, and with the use of some calculations determine that the robot will be at a certain place, but several factors contribute to this not always occurring.

The servo motors used in this project included rotary encoders so it is

known that when the motors are told to rotate X° it's known they have in fact rotated that much. This is however only half of the problem.

The next problem comes in when the static friction between the motor's rollers and the driving ball becomes less than the static friction between the ball and the ground. At this point the motors will rotate but the ball, and thus the robot, won't necessarily. So although the motors have moved the correct distance, the robot hasn't.

1.3 Other

Other things to note on the project.

The system makes use of some fairly high torque motors and contains a large amount of electronics. While the actual drain of the system is unknown, when not actively using the platform it is advised to unplug it as the battery can lose power after a few hours of only standby use.

When powered in the joystick mode the X-axis is prone to rotate very slowly. This would cause a noticeable movement on the ground and over time would include a significant drift.

When powered in the Automatic mode and waiting for locations to be programmed, both the X and Y motors rotate at a slow rate in a continuous direction. This is expected to affect locational awareness.

When monitoring the system via the visual interface on the computer, the robot is displayed on an XY grid. The position of the robot on this grid comes from the data given by the rotary encoders on the stepper motors. When in the automatic or joystick modes, the motors are moving slightly and the robot can be seen to physically move, however the visual representation of the robot doesn't move at all.

At first it was thought that this could mean the encoders results were drifting, so the motors were thus moving a little a bit the whole time to correct for this, however, when placed in non-automatic mode, where the motors are stationary, no drift on the visualisation is evident.

Further investigation revealed the problem. The PLC monitors the rate of change of the rotary encoder's value. This value is constantly changing by very small amounts, close to 0 V. This is most likely as a result of the reading of the analogue voltages for the sensor values. This is not unusual and fairly characteristic of an analogue sensor reading.

The code however is written in such a way that any change in the reading at a rate of less than 60 (units/s) is ignored. So the small drift of the sensors is ignored by the PLCs and only rapid movements, like when the robot has been told to move, are noted.

The small movement that actually does occur is due to the analogue output from the PLC to the motor controller's not always being exactly 0 V.

Another issue comes with rotation. As the ball is rotated in one direction, it is not necessarily free to do this, as the contact point with the other motor creates some resistance. This means that as the robot tries to progress in a straight line, a slight drift and rotation in one direction is evident.

For example, when travelling in a negative Y direction. the robot tends to rotate slightly in a clockwise direction and drift a bit in the negative X direction. In other applications of similar design, an omniwheel is generally used to allow the free movement in this direction.

Although when the project was created it was possible to access the controller via a Wi-Fi adaptor. Changes in the wireless infrastructure on the Reutlingen campus mean this is no longer possible. The Wi-Fi adaptor that was used is incapable of connecting to the *mycampus* Wi-Fi AP as the adaptor does not support the latest WPA security protocol. This even after the firmware on the device was updated.

Even if a connection to the AP was successful, the wireless and wired networks run on different subnets, so connection from the computer which is on the ethernet LAN would not be possible to the Robot on the Wi-Fi WLAN.

2 Wi-Fi Control

As mentioned the Wi-Fi access method was no longer available. To once again achieve this, several possibilities were put forward.

One of the easiest methods to achieve this would be to buy a Wi-Fi adaptor for the computer running the software, unfortunately the computer is running Windows XP, and to setup Windows XP as a successful wireless AP often requires testing various Wi-Fi units hoping to find one that will work correctly and allow the robot's Wi-Fi adaptor to connect to. Estimated cost as low as €15, but no guarantee of correct functioning.

If one doesn't mind disconnecting the computer from the current network, one can purchase a small Wi-Fi AP which plugs into the computer's ethernet port. The robot would then be able to connect to this. Estimated cost between €30 and €40.

Another solution which would allow the computer to remain connected to the university network and still be able to connect to the robot wirelessly would be to purchase a router unit. The university network will plug into the router and the router will NAT the devices behind it. The computer will plug into the router and the router will contain a Wi-Fi AP which the robot

can connect to. Expected cost between €40 and €60.

3 Relative Position Sensors

These are sensors that monitor relative movement of an object. That is they only track the difference in position from its original position. This means that over time errors can be added to each other and if enough time passes the object might believe it is located somewhere but is actually quite far off target.

Generally each time before the tracked object is to be used, the controller will zero itself on a known location to be as accurate as possible for as long as possible. These systems will sometimes make use of landmarks to recalibrate their locations.

3.1 Inertial navigation

Inertial navigation makes use of a collection of gyroscopes and accelerometers to monitor the movement of an object. Newton's first law states that an object will continue in a direction and at a fixed speed unless a force acts upon it (this property of a mass is known as inertia) at which point it will accelerate in the direction of the force at a rate that is proportional to the mass of the object and the magnitude of the force. $F = m \cdot a$ (Newton's 2nd law)

So if the initial variables of an object (its velocity and position) are known, and the rate of acceleration of the object can be read, it is thus possible to determine its current velocity (by integrating the acceleration values), and current position (by integrating the velocity values).

There are many different sensors available, and prices vary immensely. The most common type of device for this situation is a small XYZ 6-axis accelerometer. It detects linear acceleration in three planes as well as rotational acceleration on three axes. They are generally easily interfaced with microcontrollers via some form of serial connection.

While units like this can be picked up relatively cheaply, there are certain issues with their use. Besides the drift common in such systems, white noise and random walk also influence the signal processing that takes place and can add random errors to the data. This coupled with temperature sensitivity and calibration errors means that inertial navigation systems need a fair amount of effort to get to work adequately.

This method was partly implemented into the project by Oßmann (2009), but it was decided that due to the instability of the data received it would

not be implemented into the control system. An attempt to read this data in the robot's current state has not been performed.

3.2 Mouse sensor

Computer mice come in two forms, the old 'mechanical' ball mouse and the modern optical mouse. Both mice are able to monitor the movement of said mouse and return a change in x and y coordinates as the mouse is moved. While the technologies differ the outcome is the same.

The main downside to using a mouse as a sensor is that the mouse needs to keep almost constant contact with the moving surface. For the mobile robot in question this is not too difficult. The robot was designed to be used indoors on a flat smooth surface, similar to the desired surface for using a mouse. Outdoors on grass, dirt or uneven surfaces this method cannot be used.

Small errors may occur in reading as surfaces differ, but for the cost of a mouse a fairly accurate system can be used. To maintain constant contact between the mouse and the ground a small suspension system can be used on the mouse mounting.

Oßmann (2009) did briefly mention this method for tracking movement in his report, but the method was not one of the ones contemplated in the decisions matrix and an accelerometer was used. Although the subject has been widely covered by the hobby robotics community, no commercial robotics application has been identified that uses the method.

In most instances a Mouse sensor cannot be used by itself, as the X and Y coordinates that it feeds are relative to its orientation. So if the robot is rotated 90° then as the robot moves in the X direction, the mouse sensor would supply data that the robot had moved in the Y direction instead.

This can be countered by including some form of orientation sensor, either a compass or gyroscope capable of monitoring acceleration around an axis. Digital compasses also exist and can be used if orientation in only the horizontal plane is required.

In theory orientation for this robot shouldn't be a problem, as it is only capable of generating movement in two planes. However after testing it is evident that some undesired rotation during movement does occur as a result of system design and friction.

3.2.1 Mechanical System

Although not found very often any more, the mechanical system is very straightforward and was used exclusively during the 1990s. It works in reverse

to how the Mobile Robot in question works. A ball is always in contact with the ground. As the robot moves the ball rolls. Inside the mouse are two rollers which have slatted discs on the end. The discs are mounted between pairs of light sensors. As the discs rotate light is let through and blocked alternatively.

A simple circuit processes each pair of signals to determine how much and in what direction the discs are turning, thus producing a set of x and y coordinates for the computer to use.

3.2.2 Optical System

The optical mouse is currently the standard mouse used, whether making use of a laser or LED light source the principal remains the same. Mounted in the mouse is a tiny camera that is constantly taking pictures of the surface below it.

The mouse detects patterns in the pictures that it takes and as the mouse is moved, those patterns are tracked and a relative x and y position is processed. Different mice just use different light sources and filters to ensure the correct data is processed.

3.2.3 Obtaining data

A fair amount of work has been done in using the data from a mouse for odometry purposes. Libraries have been developed for several microcontrollers which make use of the PS2 protocol. Although most mice these days use USB, it is still possible to obtain mice which make use of PS2 and both mechanical and optical mice used this protocol.

With mechanical mice it is fairly straightforward to monitor the pulses produced by the light sensors and do the required processing oneself.

Several popular optical mouse chips have been documented well enough that the data they output can be fairly easily read using a serial connection and an appropriate microcontroller.

4 Exact Position Sensors

These systems make use of fixed tracking systems that constantly update the system to their location relative to known fixed locations. Although slight errors can occur in readings, the drift factor that can occur in the relative position systems is non-existent.

4.1 Triangulation

Although several different technologies can be used, the principal for all of them is similar. The robot would be fitted with a receiving unit which it uses to concurrently receive signals from several different base stations.

The fixed locations of all the base stations are known, and by having them transmit certain signals it is possible to calculate the robot's location relative to these base stations. This is usually done by processing the signals and comparing the time differences when receiving the signals from the different stations.

The speed at which the different signals are transmitted is known, and by looking at the difference in time it took each signal to arrive one can calculate the distance from each base to a certain accuracy. With at least three bases setup one can triangulate the position of the receiving unit.

Signals may be oscillatory, and the phase difference between the phases compared or timing data can be transmitted and this used to calculate it. This is the same principal of operation that GPS systems use. Satellites with known positions constantly transmit the time from on-board atomic clocks. GPS receivers note the time they receive the transmitted times from various satellites. The speed at which the radio transmissions travel (speed of light) is known, and from this triangulation can occur. For GPS generally at least four satellites are required before a location can be determined.

One of the cheaper methods to implement this system is to use out-of-phase ultrasonic transmissions. One of the downsides to this method is the requirement of line of sight. Although certain materials can be placed between the stations and the receiver, over short distances the change in medium can affect the accuracy of the system.

At the same time the slower speed of sound at which ultra-sonic signals travel is more desirable than the speed of light that radio and light transmissions travel at. At such high speeds, over short distances, such as in a building, only very short differences in time occur, this can require high precision, and high cost timing equipment to accurately obtain results.

In certain environments reflected signals can also disrupt the accurate tracking of a system.

While in general these systems can be very effective, setup can be a hassle. The cost of the system is also a downside and the amount of interference in most environments means it is rarely used.

4.2 Proximity

The basic idea in a proximity setup is to have an array of laser- or other proximity sensors mounted in a grid around a room. Each sensor would have an X and Y coordinate. Depending on which sensors are triggered one can determine where in the area the robot is located

The accuracy of the system is entirely reliant on how many sensors one chooses to put on each axis. The cost for high accuracy can be quite large. Also the feasibility of installing such a setup in most places means it is not often used. Furthermore it's quite easy to get interference from other objects that may enter the area. The sensors will have no way to know which object in the area is the actual object for desired detection.

A slight change to this is to have sensors only along one axis. These sensors will be capable of measuring the distance to an object, and not only if an object is present. One would be able to get X coordinates from determining which sensors are picking up an object, and Y coordinates from the distance to the object as determined by the sensors.

4.3 Inverse Proximity

Instead of placing sensors on the boundaries of an area to measure the distance to the sensor, instead two sensors can be mounted on the robot, and measure the distance to the boundary. This is a simpler method than the previous one, but is reliant on one or two aspects.

This system is incapable of measuring rotation. So if while the robot is moving it were to rotate 90°, much like the mouse sensor, the robot's coordinate system would be reversed, but this would not be known.

Another requirement for this proximity sensor setup is that of a fixed boundary. That is, long straight surfaces are required against which the reference can be drawn. This does not happen particularly often in environments where robots are used.

5 Project Changes

Due to the limited time and ease of installation it was suggested that the Inverse Proxy solution for one axis be attempted. The sensor was to be integrated into the current system. It would be required that the robot be able to move along a specific axis, while maintaining a constant distance from the wall on the other, sensor-monitored axis.

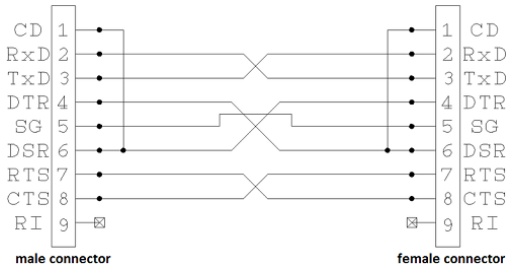

Figure 1: Null-Modem Serial Cable Wiring (MODEM REVIEWS 2012)

5.1 Wireless Connection

On attempting to achieve wireless communication and during the process of changing network settings the PLC became unresponsive. Any further attempts to connect to the PLC via ethernet failed. Furthermore the PLC would not go into RUN mode, it is uncertain why this happened but it appears to be as a direct result of the network settings being changed, even though they shouldn't influence the normal operating of the PLC.

The backup solution to connecting to the PLC requires the use of a Null-Modem Serial cable and the setting up of a PPP network connection from a computer. A null-modem serial cable had to be made up, the cable is similar to a standard serial cable, but that the TX and RX lines are switched. A diagram for such a cable can be seen in figure 1. This cable was added to the box of hardware relating to the Mobile Robot.

Instructions for connecting to the PLC in this manner can be found both in the the help file for the PLC and in the thesis by Oßmann (2009). After connecting to the PLC via this method and resetting the network settings, it was again possible to connect to to the the PLC via standard ethernet connection.

As mentioned earlier the Wi-Fi adapter originally allowed the PLC's ethernet port to act as a Wi-Fi device and connect to an established Wi-Fi AP, but this was no longer possible. Several suggestions were made earlier in this report, but after further investigation of the Wi-Fi adaptor being used on the robot, an alternative solution was found.

It was found that the adaptor could be setup in several different modes, and one of these modes was as a Wi-Fi AP. In this way any Wi-Fi device would be able to connect to the AP, which effectively shares the PLC's network.

The PLC was given a static IP address of 192.168.1.71. The Wi-Fi adaptor has a default IP address of 192.168.1.1 which was kept. It was then setup in Wi-Fi AP mode, using WPA2 protection with the password *asdfghjk*.

note: When connecting to the PLC directly via ethernet, the same address must be used as if it were via the Wi-Fi. The addresses specified in the previous theses are no longer valid.

An Asus WL-167G V3 USB Wi-Fi unit was purchased for the computer running the CoDeSys PLC interface. It was installed and given a manually defined static IP address of 192.168.1.35. The connection settings in CoDeSys also had to be altered to reflect the new addresses. Opening the connection configuration options, the IP address for the *Berghof-WiFi* connection was changed to 192.168.1.71 to reflect the new network configuration.

5.2 Motor Drift

As was mentioned previously, whenever the robot is in automatic mode or joystick mode the motors tend to move at a very small speed. It was discovered that this was because there was a slight difference between the 0 V output signal from the PLC and the 0 V that the motor controller recognised.

That is the motor controller read the 0 V from the PLC as slightly above or below 0 V, and thus thought it was being asked to drive the motors at a very low speed. One of the previous people who worked on the project tried to solve this by setting the 0 speed value for the motor speed variable on the PLC to a value that the motor controller would recognise as 0. This works for some time, but as different batteries are used and the battery loses power this too seems to drift.

The motor controller itself also has an offset terminal on it, which can be adjusted with a screw driver. Setting this so the motors didn't move worked at first, but after time this too seemed to drift.

5.3 IR Sensor

The first sensor that was tested was a Sharp GP2D12. It has a non-linear analogue voltage output between 0 and 2.6 V (visible in figure 2), requires a 5 V supply and has a detection area between 6 and 80 cm (Sharp 2012).

The sensor was mounted on the side of the sensor tower that was added during Oßmann (2009) project. A hole was drilled into the side of the flat panel that holds the Wi-Fi adaptor in place and the sensor was mounted vertically, as per the specification for horizontal motion. Power was obtained from the 5 V DC-DC transformer that is used to provide power to the Wi-Fi Adaptor.

Pins were added to the one side of the board and linked to the output of the converter as well as to a set of screw terminals on the other side of the

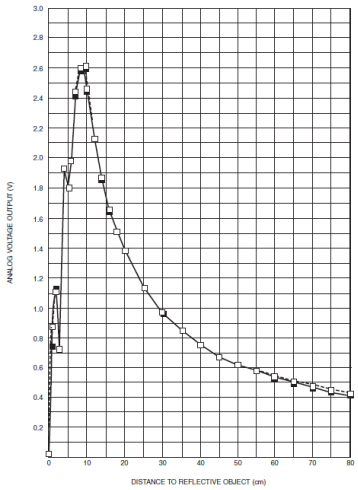

Figure 2: Graph indicating the Voltage Output vs Distance from sensor (Sharp 2012)

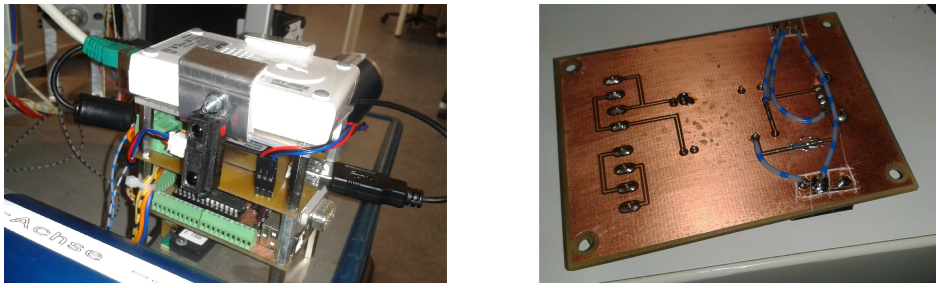

Figure 3: Modifications - left: mounted sensor, right: PCB modifications

board. This aided in the installing of the unit. Images of the modifications can be seen in figure 3.

Two wires, a positive and negative signal were then laid to the PLC. A negative was required as the ground line on the DC-DC converter was isolated from the system's ground. The wires were connected to analogue input 6 on the PLC. The input from the port could then be read via variable *%IW7*.

A new function block *Position_Proxy* was created to handle the values. A variable *Y_proxy_dist* was declared which read the value from the analogue input and converted it to a voltage value. This data was then processed by a series of *IF* statements to calculate the relevant distance. The statements were drawn up using linear interpolation values from the graph in figure 2.

Although this method worked effectively, it was found that the sensor's output varied fairly erratically, by up to 10cm at times. This was deemed to

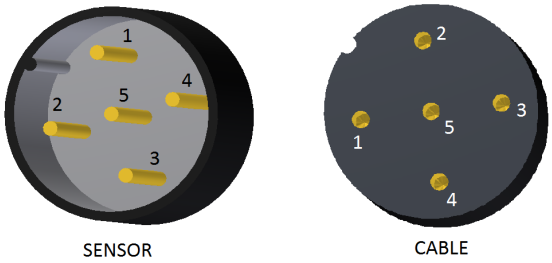

Figure 4: Wiring diagram for sensor cable

Pin #	Colour	Function
1	Brown	18-30 V DC +
2	White	!Enable
3	Blue	Ground
4	Black	Switching Out
5	Grey	4-20mA Out

Table 1: Information for Sensor Wiring

be due to the design of the sensor. For the purposes of this project it was felt this would not allow for effective control. The IR sensor was subsequently replaced.

5.4 Laser Sensor

In place of the IR sensor, another sensor became available. A Leuze Electronic ODSL9/C6 laser sensor. The sensor is designed for industrial settings, has an LCD display, is powered by 18-30 V DC, has both an analogue current output (4-20 mA) and a PNP switching output. The one downside is the measurement range which is only from 5-10 cm, but it has a resolution of 0.01mm and repeatability of $\pm 0.25\%$ (LEUZE ELECTRONIC 2009).

The sensor makes use of a standard M12, 5-pin industrial connector. However only 4-pin connectors could be located. The missing pin was the one that provides the analogue current output. As such a 4-pin connector was modified to support a fifth pin. A hole was drilled in the centre of the plug, and another hole perpendicularly from the side. A wire was laid through this hole with a cylindrical metal sheath being soldered to the wire and placed in the centre hole.

A wiring diagram for a standard plug and for this sensor can be seen in figure 4 and table 1.

Only three wires were required. Positive 24 V and the ground line were

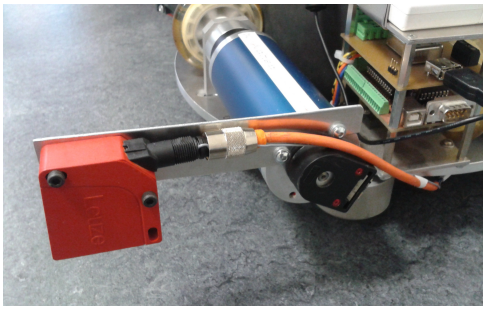

Figure 5: Sensor on manufactured mounting

taken from the main distribution clamps, and the third wire connected to the analogue output of the sensor was laid to the sixth analogue input on the PLC, the same as used with the IR sensor.

A mounting for the sensor also had to be created. Space was found on the end of the motors where two threaded holes were. A piece of 3 mm Aluminium was taken and cut to size, with appropriate holes drilled into it. The sensor was mounted on one end of the aluminium, and the other end was bolted to the back of the motor. The mounted sensor can be seen in figure 5.

The sensor outputs an analogue current relating to the distance, however the PLC had only analogue voltage inputs. As such a 470 Ω resistor was put in between the line and ground, and a wire was taken from the above the resistor to the analogue input. This would allow the 4-20 mA signal to use almost the entire range of the 0-10 V analogue voltage input on the PLC.

To calibrate the sensor, values on the sensor's LCD were compared to the values read by the PLC. From this a set of linear sections of readings was determined. The PLC code used for the IR sensor was modified to be applicable to the laser sensor by using the generated graph that can be seen in figure 6.

The datasheet for the sensor declared that the output current can be deemed linear during Series2; and Series1 and Series3 are assumed linear for their respective areas. Although the measuring range of the sensor was relatively small, the high accuracy and stability of the measured distance meant it would be more effective at being used for controlling movement.

5.5 Visual Alterations

To simplify the use of the sensor and the updating of the Robot's control method, several changes were made to the CodeSys interface developed by Oßmann (2009). In the object *PLC_VISU*, in the bottom right hand corner,

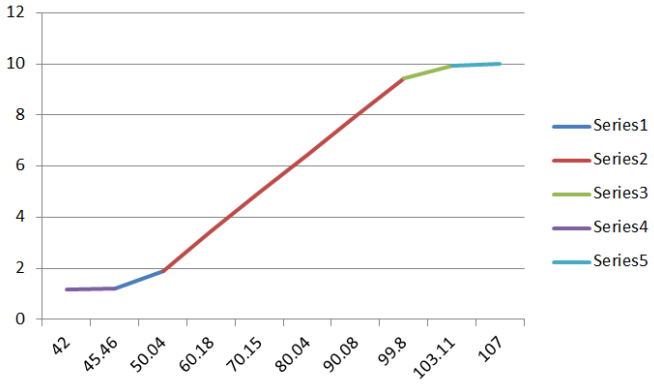

Figure 6: Generate Calibration Graph

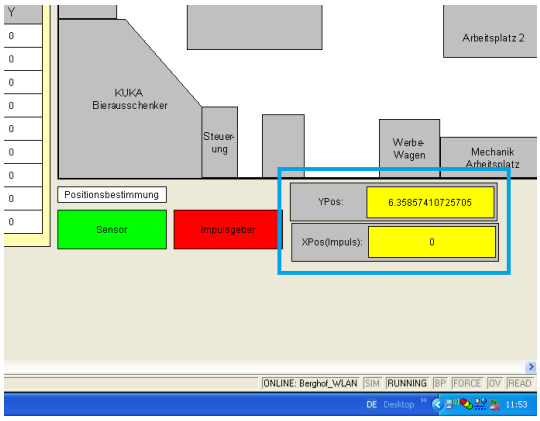

Figure 7: Screenshot of Visual Alterations to Interface

two extra values were added (see figure 7). The values relate to the X and Y coordinates of the Robot. The X value given is always as calculated from the rotary encoder. The Y value will either show the sensor reading or the value calculated from the rotary encoder, depending on which button is selected to the left of these values; *Sensor* or *impulsgeber*. Both values are given in centimetres.

The second addition is the *Control_Graphs* Object. It shows two graphs (see figure 8) relating to the X (top) and Y (bottom) movement of the robot. The black line on each graph refers to the desired location of the robot (internal variables *X_Soll* and *Y_Soll*). The red line indicates the position of the robot at that moment. On the X graph this will always be the value from the rotary encoders. On the Y graph this will vary between the rotary encoder value and the sensor value, depending on what is selected on the visual interface object.

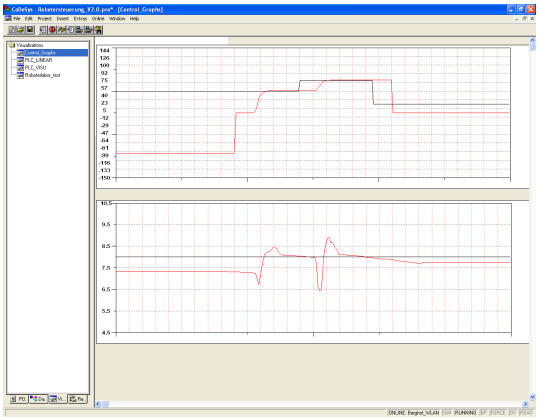

Figure 8: Control Graphs that were added

5.6 Control Alterations

The next step was to alter the way the Robot controlled its motion. The original programming made use of the *Linearfahrt* (Linear Drive) function block to move the robot. This function compared the current location to the desired location and calculated the average speed that had to done for each of the X and Y axes, so that the robot would travel in a straight line to the next point.

The movement was then split up into three sections for both the X and Y movements. An acceleration, a constant speed, and a deceleration section. For each section the programming made use of the earlier calculated values to carefully provide linear acceleration and movement from point A to point B.

Although effective, the movement basically relied on an open loop system, comparing values to the rotary encoder to ensure 'movement' was proceeding as expected. With the sensor now installed to provide accurate measurements, a closed loop control system could be utilised.

The CodeSys environment has a built in PID controller and it was felt that this would be the best way to proceed with the control of the system. The previous *Linearfahrt* function block was modified to suit the new design. All the previous control instructions were removed to avoid confusion, and the new, simpler PID controllers were added. One was added for both the X and Y controllers.

With the PID controllers instantiated and functioning, it was necessary to tune them. With the D multiplier set to 0 and the I multiplier set to 1, the P multiplier was altered till stable control was established. From there the I multiplier was brought in. This tuning took place using only the rotary

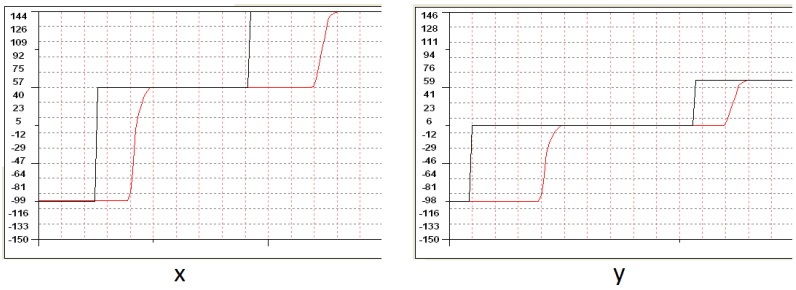

Figure 9: Graph indicating PID control

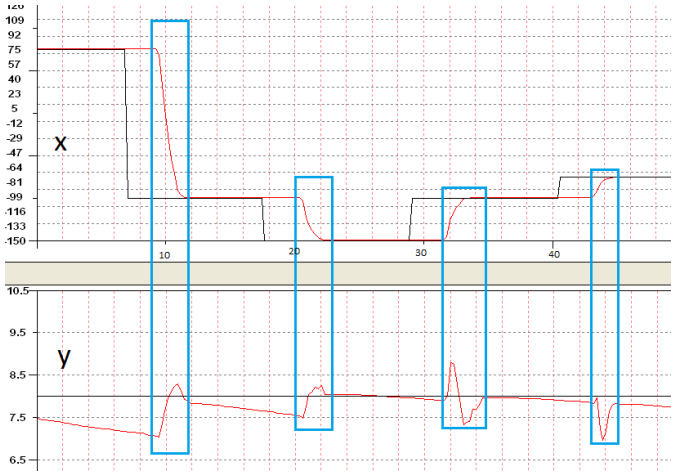

Figure 10: Graph reflecting successive robot movements

encoder values, and without the driving ball being present. The resulting control graphs can be seen in figure 9.

The second control graph was then adapted to display only values between 4.5 and 10.5 cm. The measuring range of the sensor and some testing was run with the robot moving in the X direction while maintaining a fixed Y position. Drift occurs when the robot stops due to the earlier mentioned, slight difference between the motor controller's ground level and the 0 V analogue output from the PLC.

Figures 10 and 11 show the results of several successive movements of the robot. The blue rectangles indicate times when the robot was commanded to move from one position to a new position. Each movement took between 2 and 3 seconds. The Y values are those recorded from the sensor. Although movement in the Y axis doesn't look smooth, the correct position is always achieved.

The jagged motion can be attributed to the rotation and other non-

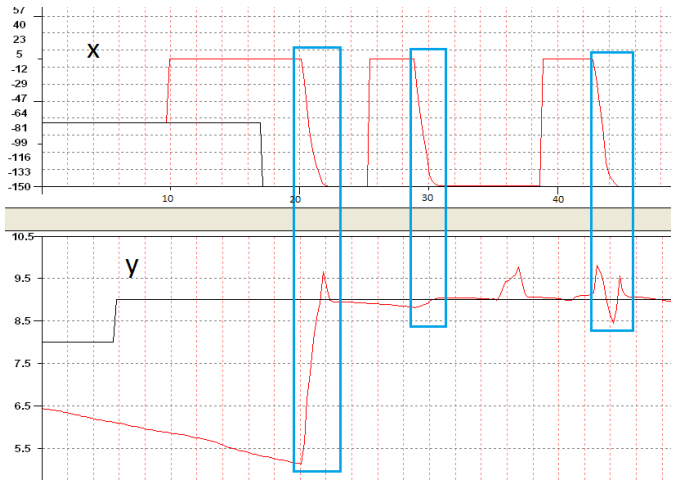

Figure 11: Graph reflecting successive robot movements

modelled movements due to friction in the movement of the robot.

A copy of the original code can be found on the desktop of the *Besucher* login account on the appropriate computer. *Robotersteuerung - 00 original*. Another copy can be found on the CD as part of Oßmann (2009)'s thesis.

The updated code can also be found on the desktop of the *Besucher* account. Stored in folder the folder *Robotersteuerung - 03 proxy with PID*. Another copy can be found on the web at http://gcawood.com/files/roboter.zip.

5.7 Control Problem

One problem that was noted with the inclusion of the PID controllers, is that when a distance of 200 or more needs to be covered, the PID controller wants to accelerate the motors fairly fast, which causes the current limiter on the motors to kick in.

An attempt to deal with this resulted in lowering the P value of the controller significantly whenever the distance to be travelled was greater than 200. This however lead to a larger overshoot on the desired location, no amount of tuning seemed able to solve this problem.

5.8 How to control

Although changes have been made to the way the control of movement works on the robot, the basic process to start movements hasn't changed from the previous projects and a guide can be found in Oßmann (2009)'s thesis. The basic process is as follows:

- Create connection to robot from CodeSys.
- Open the *PLC_VISU* Object.
- At the bottom of the window select if you want to use the analogue sensor (*Sensor*), or the rotary encoders (*Impulsgeber*) to determine the robot's position.
- Select whether you want to make one linear movement (*Einzel Bahnfart*), or several successive movements (*Automatische Bahnfart*).
- Fill in the required values and click the appropriate start (*-starten*) button

  Clicking *Reset* at any time will stop movement and reset the rotary encoder sensor values to zero.

6 Conclusion

After investigating different methods with which to monitor the movement of an omnidirectional robot a method making use of proximity sensors mounted on the robot to note it's position to a surrounding wall was tested. Two separate sensors were tested. The first infra-red sensor didn't provide the required accuracy or stability for the project. It was replaced with a more accurate laser proximity sensor. Although the sensor had a limited range the sensor was suitable for testing.

A Wi-Fi connection to the robot was re-established by setting up a Wi-Fi AP on the robot and connecting to it via a USB Wi-Fi adaptor on the desktop PC. Several modifications were also made to the visual interface for the robot.

To achieve better control of the robot, the movement controlling code was rewritten. The initial code looked at the current and required positions and performed a direct movement to the point using a linear acceleration and deceleration method. This was replaced with PID controllers for both the X and Y axes. Some tuning was performed for optimal movement and successful movement of the robot was achieved.

It is suggested that further testing be performed with a sensor with a greater range, as well as placing a second sensor on the X axis.

References

Leuze Electronic, 2009. *ODS... 9 / ODS... 96B - Optical distance sensors*. Leuze Electronic.

Modem Reviews, September 2012. Null Modem Wiring. internet: http://modemzz.com/null-modem-wiring/.

Christoph Oßmann, July 2009. *Navigations, Steuerungs- und Kommunikationsfunktionen für eine mobile Roboterplattform hoher Beweglichkeit*. Mechatronic bachelor's project report, Reutlingen university, Stockacherstr 5 72810 Gomaringen.

Sharp, 2012. *Sharp GP2D12 Optoelectronic Device*. Sharp.

Hanna Weiß, January 2009. *Konzeption unf Implementierung von Stteuerprogrammen für eine mobile Roboterplattform hoher Beweglichkeit sowie Optimeierung des Gesamtsystems*. Mechatronic bachelors project report, Reutlingen University, Necharhalde 74 72108 Rottenburg.