Video: Figure 8

Start playback

Örebro Unversity
School of Science and Technology

Örebro University

Project: Differential Drive via WiFi

Project presentation

Benny Frost, Tom Olsson

Örebro University

January 11, 2016

January 11, 2016

© Benny Frost, Tom Olsson

Outline

Project definition

- Build a robot with two motors

Project definition

- Build a robot with two motors
- Setup the two motors, along with a differential drive controller

Project definition

- Build a robot with two motors
- Setup the two motors, along with a differential drive controller
- Add WiFi communication instead of serial

Project definition

- Build a robot with two motors
- Setup the two motors, along with a differential drive controller
- Add WiFi communication instead of serial
- Calibrate the odometry

Project definition

- Build a robot with two motors
- Setup the two motors, along with a differential drive controller
- Add WiFi communication instead of serial
- Calibrate the odometry

Hardware: Arduino Due [1], Motor Shield rev. 3 [2], WiFi Shield [3]

Outline

Building the robot

- Main body built from aluminum plating and profiles

Building the robot

- Main body built from aluminum plating and profiles
- Details primarily built from plexiglass

Building the robot

- Main body built from aluminum plating and profiles
- Details primarily built from plexiglass
- Two driving front wheels

Building the robot

- Main body built from aluminum plating and profiles
- Details primarily built from plexiglass
- Two driving front wheels
- First design:

Building the robot

- Main body built from aluminum plating and profiles
- Details primarily built from plexiglass
- Two driving front wheels
- First design:
  - Two slippery/smooth rear wheels

Building the robot

- Main body built from aluminum plating and profiles
- Details primarily built from plexiglass
- Two driving front wheels
- First design:
  - Two slippery/smooth rear wheels

  $\Rightarrow$ Motors too weak to force the sliding

Building the robot

- Main body built from aluminum plating and profiles
- Details primarily built from plexiglass
- Two driving front wheels
- First design:
  - Two slippery/smooth rear wheels

  $\Rightarrow$ Motors too weak to force the sliding
- Second design:

Building the robot

- Main body built from aluminum plating and profiles
- Details primarily built from plexiglass
- Two driving front wheels
- First design:
  - Two slippery/smooth rear wheels

  $\Rightarrow$ Motors too weak to force the sliding
- Second design:
  - One centered swivel wheel at the rear

Building the robot

- Main body built from aluminum plating and profiles
- Details primarily built from plexiglass
- Two driving front wheels
- First design:
  - Two slippery/smooth rear wheels

  $\Rightarrow$ Motors too weak to force the sliding
- Second design:
  - One centered swivel wheel at the rear

  $\Rightarrow$ Much better performance, but some issues because of the swivel (shopping cart syndrome)

Motors & power

- The same motor shield as for laboration one was used, the Motor Shield Rev. 3 [2]

Motors & power

- The same motor shield as for laboration one was used, the Motor Shield Rev. 3 [2]
- The shield can handle up to 18 V input, and output 2 A per channel, but this requires the Arduino power circuits (3.3 V, 5 V and *vin*) to be completely separated from the motor power supply

Motors & power

- The same motor shield as for laboration one was used, the Motor Shield Rev. 3 [2]
- The shield can handle up to 18 V input, and output 2 A per channel, but this requires the Arduino power circuits (3.3 V, 5 V and *vin*) to be completely separated from the motor power supply
- To supply it with power, we created a battery pack with 12 AA batteries in series, which supply 18 V total

Motors & power

- The same motor shield as for laboration one was used, the Motor Shield Rev. 3 [2]
- The shield can handle up to 18 V input, and output 2 A per channel, but this requires the Arduino power circuits (3.3 V, 5 V and *vin*) to be completely separated from the motor power supply
- To supply it with power, we created a battery pack with 12 AA batteries in series, which supply 18 V total
- The pins on the Arduino main board that were needed for the WiFi card were moved to other pins

Motors & power

- The same motor shield as for laboration one was used, the Motor Shield Rev. 3 [2]
- The shield can handle up to 18 V input, and output 2 A per channel, but this requires the Arduino power circuits (3.3 V, 5 V and *vin*) to be completely separated from the motor power supply
- To supply it with power, we created a battery pack with 12 AA batteries in series, which supply 18 V total
- The pins on the Arduino main board that were needed for the WiFi card were moved to other pins
- Similarly, the main Arduino board was battery powered by a 9 V battery

Outline

Tuning & setup

- Two different PID-setups, one for each motor

Tuning & setup

- Two different PID-setups, one for each motor
- One big problem to solve:

Tuning & setup

- Two different PID-setups, one for each motor
- One big problem to solve:
  - The motors have a large (but different) deadband where it doesn't move

Tuning & setup

- Two different PID-setups, one for each motor
- One big problem to solve:
  - The motors have a large (but different) deadband where it doesn't move

  $\Rightarrow$ The deadband roughly in the range $[0, 700]$, but one motor is slightly easier to start

Tuning & setup

- Two different PID-setups, one for each motor
- One big problem to solve:
  - The motors have a large (but different) deadband where it doesn't move

  $\Rightarrow$ The deadband roughly in the range $[0, 700]$, but one motor is slightly easier to start
- To reduce the impact of the deadband, all actuation values from 10 to 700 were set to 700, and all below 10 were set to 0

Tuning & setup

- Two different PID-setups, one for each motor
- One big problem to solve:
  - The motors have a large (but different) deadband where it doesn't move

  $\Rightarrow$ The deadband roughly in the range $[0, 700]$, but one motor is slightly easier to start
- To reduce the impact of the deadband, all actuation values from 10 to 700 were set to 700, and all below 10 were set to 0
- We also added feedback in the PID controller, so that some of the current output PWM is counted as "promised speed", which smoothes the PID output by approximating the smoothing applied to the velocity reading

Outline

Adding the WiFi communication

- The `ros_lib` `NodeHandle` is a template class, and accepts a `Stream` interface object as the template argument

Adding the WiFi communication

- The `ros_lib` `NodeHandle` is a template class, and accepts a `Stream` interface object as the template argument
- This is normally a `Serial` class reference

Adding the WiFi communication

- The `ros_lib` `NodeHandle` is a template class, and accepts a `Stream` interface object as the template argument
- This is normally a `Serial` class reference
- However, the `WiFi` module is also a derivative of `Stream`

Adding the WiFi communication

- The `ros_lib` `NodeHandle` is a template class, and accepts a `Stream` interface object as the template argument
- This is normally a `Serial` class reference
- However, the `WiFi` module is also a derivative of `Stream`
- Therefore, it should be relatively easy to change between the two classes

Adding the WiFi communication

- The `ros_lib` `NodeHandle` is a template class, and accepts a `Stream` interface object as the template argument
- This is normally a `Serial` class reference
- However, the `WiFi` module is also a derivative of `Stream`
- Therefore, it should be relatively easy to change between the two classes
  - Replace references to `Serial` with `WifiClient`

Adding the WiFi communication

- The `ros_lib` `NodeHandle` is a template class, and accepts a `Stream` interface object as the template argument
- This is normally a `Serial` class reference
- However, the `WiFi` module is also a derivative of `Stream`
- Therefore, it should be relatively easy to change between the two classes
  - Replace references to `Serial` with `WifiClient`
  - Add connection management code

Adding the WiFi communication

- The `ros_lib` `NodeHandle` is a template class, and accepts a `Stream` interface object as the template argument
- This is normally a `Serial` class reference
- However, the `WiFi` module is also a derivative of `Stream`
- Therefore, it should be relatively easy to change between the two classes
  - Replace references to `Serial` with `WifiClient`
  - Add connection management code
  - *Done?*

Arduino WiFi

- The official WiFi shield from Arduino [3]

Arduino WiFi

- The official WiFi shield from Arduino [3]
- In order to connect it to the main board some pins had to be rewired, as some are used by both the Motor shield and the WiFi shield

Arduino WiFi

- The official WiFi shield from Arduino [3]
- In order to connect it to the main board some pins had to be rewired, as some are used by both the Motor shield and the WiFi shield
- Comes with several drawbacks that we discovered on our own and verified with other sources:

Arduino WiFi

- The official WiFi shield from Arduino [3]
- In order to connect it to the main board some pins had to be rewired, as some are used by both the Motor shield and the WiFi shield
- Comes with several drawbacks that we discovered on our own and verified with other sources:
  - Has a 2-second cycle rate for transmitting - in practice, it only sends packages once every other second

Arduino WiFi

- The official WiFi shield from Arduino [3]
- In order to connect it to the main board some pins had to be rewired, as some are used by both the Motor shield and the WiFi shield
- Comes with several drawbacks that we discovered on our own and verified with other sources:
  - Has a 2-second cycle rate for transmitting - in practice, it only sends packages once every other second
  - Has a maximum message size of 92 bytes for each call to the shield firmware - a limitation in the SPI bus

Arduino WiFi

- The official WiFi shield from Arduino [3]
- In order to connect it to the main board some pins had to be rewired, as some are used by both the Motor shield and the WiFi shield
- Comes with several drawbacks that we discovered on our own and verified with other sources:
  - Has a 2-second cycle rate for transmitting - in practice, it only sends packages once every other second
  - Has a maximum message size of 92 bytes for each call to the shield firmware - a limitation in the SPI bus
  - Will silently fail on any overflow in any buffer [4, 5]

Arduino WiFi

- The official WiFi shield from Arduino [3]
- In order to connect it to the main board some pins had to be rewired, as some are used by both the Motor shield and the WiFi shield
- Comes with several drawbacks that we discovered on our own and verified with other sources:
  - Has a 2-second cycle rate for transmitting - in practice, it only sends packages once every other second
  - Has a maximum message size of 92 bytes for each call to the shield firmware - a limitation in the SPI bus
  - Will silently fail on any overflow in any buffer [4, 5]
- Because of these problems, eventually we replaced the WiFi shield with an ESP8266 microcontroller as WiFi card

ESP8266

- The ESP8266 is a standalone microcontroller with built-in WiFi [6]

ESP8266

- The ESP8266 is a standalone microcontroller with built-in WiFi [6]
- This card is often recommended instead of the official WiFi card both because of its cost as well as good performance

ESP8266

- The ESP8266 is a standalone microcontroller with built-in WiFi [6]
- This card is often recommended instead of the official WiFi card both because of its cost as well as good performance
- Has its own toolchain, and runs the code on its own in a separate process from the main Arduino board **asynchronously**

ESP8266

- The ESP8266 is a standalone microcontroller with built-in WiFi [6]
- This card is often recommended instead of the official WiFi card both because of its cost as well as good performance
- Has its own toolchain, and runs the code on its own in a separate process from the main Arduino board **asynchronously**
- In this case, it is used like a router, only relaying information between two communicating devices

Outline

Communication

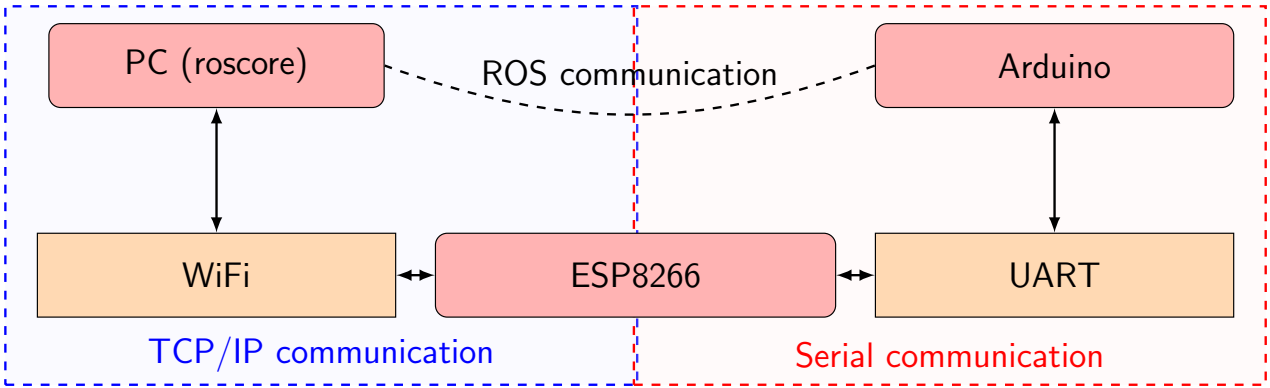

Figure: System view of communication

PC setup

- The PC runs the `roscore` for the system

PC setup

- The PC runs the `roscore` for the system
- Two nodes are used on the PC to control the robot:

PC setup

- The PC runs the `roscore` for the system
- Two nodes are used on the PC to control the robot:
  - `rosserial_python` in TCP/IP mode to handle communication and serialization

PC setup

- The PC runs the `roscore` for the system
- Two nodes are used on the PC to control the robot:
  - `rosserial_python` in TCP/IP mode to handle communication and serialization
  - a Python script for sending command messages to the robot (`geometry_msgs::Twist`). This script comes from Husqvarna and is intended for research with their robotic mower

ESP8266 setup

- The ESP8266 is set up to be a very basic passthrough module

ESP8266 setup

- The ESP8266 is set up to be a very basic passthrough module
- It does two things:

ESP8266 setup

- The ESP8266 is set up to be a very basic passthrough module
- It does two things:
  - Read from WiFi, write to Serial

ESP8266 setup

- The ESP8266 is set up to be a very basic passthrough module
- It does two things:
  - Read from WiFi, write to Serial
  - Read from Serial, write to WiFi

ESP8266 setup

- The ESP8266 is set up to be a very basic passthrough module
- It does two things:
  - Read from WiFi, write to Serial
  - Read from Serial, write to WiFi
- It also makes sure that the WiFi connection is maintained

Arduino setup

- The Arduino module is almost the same as laboration 1, but is a differential drive controller instead of a basic velocity controller

Arduino setup

- The Arduino module is almost the same as laboration 1, but is a differential drive controller instead of a basic velocity controller
- Three changes were made:

Arduino setup

- The Arduino module is almost the same as laboration 1, but is a differential drive controller instead of a basic velocity controller
- Three changes were made:
  - Reduce the serial bus speed by 25 % down to 38.4k

Arduino setup

- The Arduino module is almost the same as laboration 1, but is a differential drive controller instead of a basic velocity controller
- Three changes were made:
  - Reduce the serial bus speed by 25 % down to 38.4k
  - Add a “retry” for delayed bytes allowing up to 20ms of packet delays

Arduino setup

- The Arduino module is almost the same as laboration 1, but is a differential drive controller instead of a basic velocity controller
- Three changes were made:
  - Reduce the serial bus speed by 25 % down to 38.4k
  - Add a “retry” for delayed bytes allowing up to 20ms of packet delays
  - Use Serial1 (TX1,RX1) instead of Serial/USB (TX0,RX0)

Arduino setup

- The Arduino module is almost the same as laboration 1, but is a differential drive controller instead of a basic velocity controller
- Three changes were made:
  - Reduce the serial bus speed by 25 % down to 38.4k
  - Add a “retry” for delayed bytes allowing up to 20ms of packet delays
  - Use Serial1 (TX1,RX1) instead of Serial/USB (TX0,RX0)
- And as mentioned before, a feedback term was added to the PID controller

Reason for changes: protocol mismatch

TCP/IP

Serial

Reason for changes: protocol mismatch

TCP/IP

- Guaranteed delivery, but not in order and not at specific time point

Serial

- Guaranteed order and specific time-point

Reason for changes: protocol mismatch

TCP/IP

- Guaranteed delivery, but not in order and not at specific time point
- Varying speed

Serial

- Guaranteed order and specific time-point
- Fixed speed (baud rate)

Reason for changes: protocol mismatch

TCP/IP

- Guaranteed delivery, but not in order and not at specific time point
- Varying speed
- Any packet size

Serial

- Guaranteed order and specific time-point
- Fixed speed (baud rate)
- Fixed packet size

Reason for changes: protocol mismatch

TCP/IP

- Guaranteed delivery, but not in order and not at specific time point
- Varying speed
- Any packet size

Serial

- Guaranteed order and specific time-point
- Fixed speed (baud rate)
- Fixed packet size

$\Rightarrow$ These do not match very well, which causes synchronization problems

Reason for changes: protocol mismatch

TCP/IP

- Guaranteed delivery, but not in order and not at specific time point
- Varying speed
- Any packet size

Serial

- Guaranteed order and specific time-point
- Fixed speed (baud rate)
- Fixed packet size

$\Rightarrow$ These do not match very well, which causes synchronization problems

$\Rightarrow$ Generally because the Serial is too fast for the WiFi communication

Reason for changes: protocol mismatch

TCP/IP

- Guaranteed delivery, but not in order and not at specific time point
- Varying speed
- Any packet size

Serial

- Guaranteed order and specific time-point
- Fixed speed (baud rate)
- Fixed packet size

⇒ These do not match very well, which causes synchronization problems

⇒ Generally because the Serial is too fast for the WiFi communication

⇒ This causes the Arduino to receive "empty" from the Serial bus, while the ESP8266 is still processing parts of the message from the WiFi connection

Reason for changes: protocol mismatch

TCP/IP

- Guaranteed delivery, but not in order and not at specific time point
- Varying speed
- Any packet size

Serial

- Guaranteed order and specific time-point
- Fixed speed (baud rate)
- Fixed packet size

⇒ These do not match very well, which causes synchronization problems

⇒ Generally because the Serial is too fast for the WiFi communication

⇒ This causes the Arduino to receive "empty" from the Serial bus, while the ESP8266 is still processing parts of the message from the WiFi connection

⇒ When reading empty, the Arduino will (by default) drop the message entirely and ask for a resynchronization... which will fail too

Outline

Odometry measurements

- Very random behaviour; and very inaccurate when doing "goto" commands. These were added as an afterthought, and this may affect their performance.

Odometry measurements

- Very random behaviour; and very inaccurate when doing "goto" commands. These were added as an afterthought, and this may affect their performance.
- The rear wheel has serious problems with movement and will often get stuck at odd angles causing the robot to drift

Odometry measurements

- Very random behaviour; and very inaccurate when doing "goto" commands. These were added as an afterthought, and this may affect their performance.
- The rear wheel has serious problems with movement and will often get stuck at odd angles causing the robot to drift
- Generally, the first line of a square will be perfectly straight, and then after the first turn it will keep turning slightly without θ reflecting this

Odometry measurements

- Very random behaviour; and very inaccurate when doing "goto" commands. These were added as an afterthought, and this may affect their performance.
- The rear wheel has serious problems with movement and will often get stuck at odd angles causing the robot to drift
- Generally, the first line of a square will be perfectly straight, and then after the first turn it will keep turning slightly without θ reflecting this
- However, when manually controlling the robot and sending velocity commands it is much easier to get accurate readings

Odometry measurements

- Very random behaviour; and very inaccurate when doing "goto" commands. These were added as an afterthought, and this may affect their performance.
- The rear wheel has serious problems with movement and will often get stuck at odd angles causing the robot to drift
- Generally, the first line of a square will be perfectly straight, and then after the first turn it will keep turning slightly without θ reflecting this
- However, when manually controlling the robot and sending velocity commands it is much easier to get accurate readings
- By doing this, it is easy to see that the error is introduced each time a turn is made, and that this error is caused by the swivel wheel going the wrong way or getting stuck when starting to move forward after the turn

Video: Wheel demonstrations

Start playback

Arduino LLC, "Arduino - arduino due." https://www.arduino.cc/en/Main/ArduinoBoardDue. (Last Visited on 01/10/2016).

Arduino LLC, "Arduino - arduino motor shield." https://www.arduino.cc/en/Main/ArduinoMotorShieldR3. (Last Visited on 01/10/2016).

Arduino LLC, "Arduino - arduino wifi shield." https://www.arduino.cc/en/Main/ArduinoWiFiShield. (Last Visited on 01/10/2016).

MattS-UK, "Arduino wificlient (tcp)." http://mssystems.emscom.net/helpdesk/knowledgebase.php?article=51, November 2013. (Last visited on 01/10/2016).

Group discussion, "Wifi shield tcp to webserver is very slow." http://forum.arduino.cc/index.php?topic=123824.0, September 2012. (Last visited on 01/10/2016).

NURDSpace, "Esp8266 - nurdspace." http://nurdspace.nl/ESP8266, November 2014. (Last visited on 01/10/2016).

