Autonomous driving using an RC car with UWB positioning

Final Presentation, **Project in Embedded Systems 1TE721**

Jakob Nyberg
Torun Nicander

Today

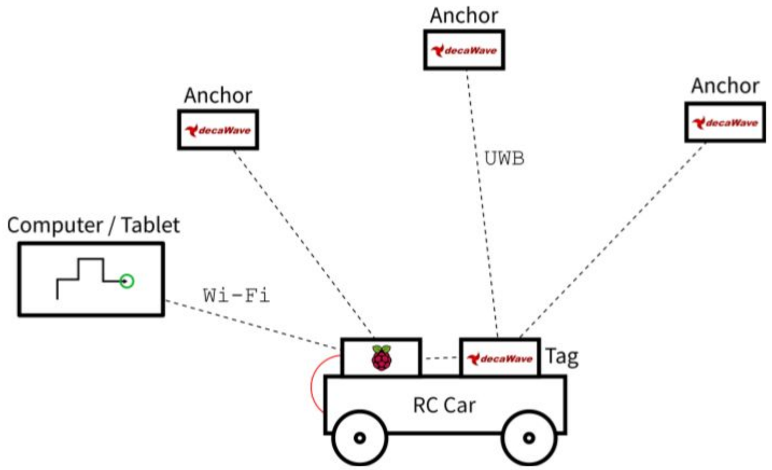
1. Introduction
    a. Objective
    b. System Overview
2. Demonstration
3. Hardware
4. Important concepts
    a. Ultra wideband signals
    b. Kalman Filter
    c. Pathing
    d. Pure Pursuit
5. Software
6. Grading Criteria

1. Introduction

1a. Objective

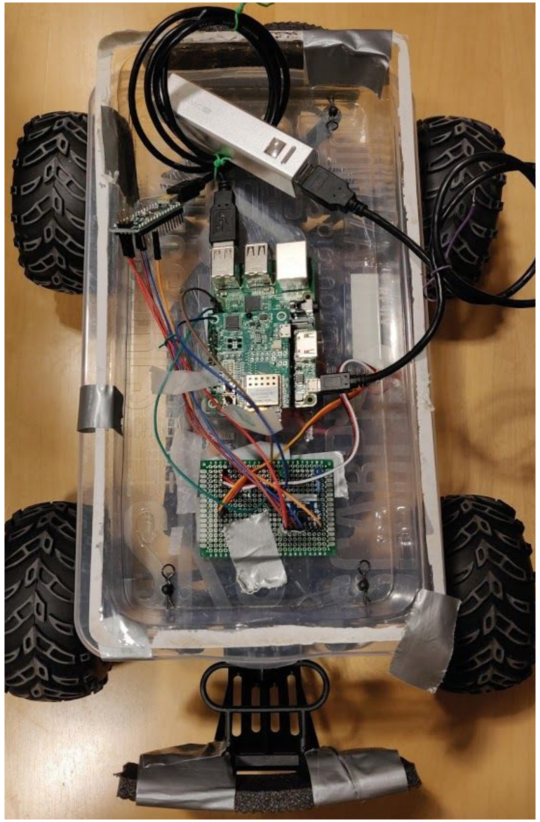
- A car that can drive itself along a path generated by user specified points in an indoor environment
- An ultra wideband real time location system is used for location estimation and tracking
- A web interface to control car and view data.

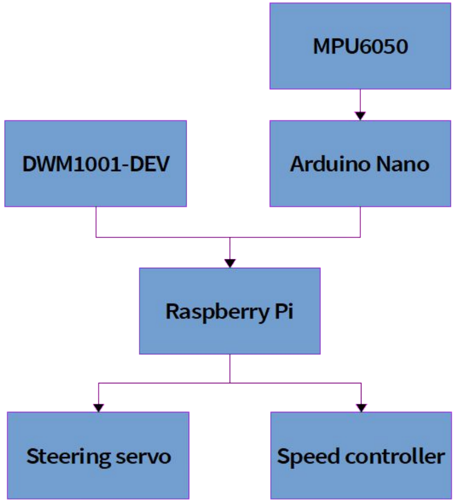
1b. System Overview



2. Demonstration

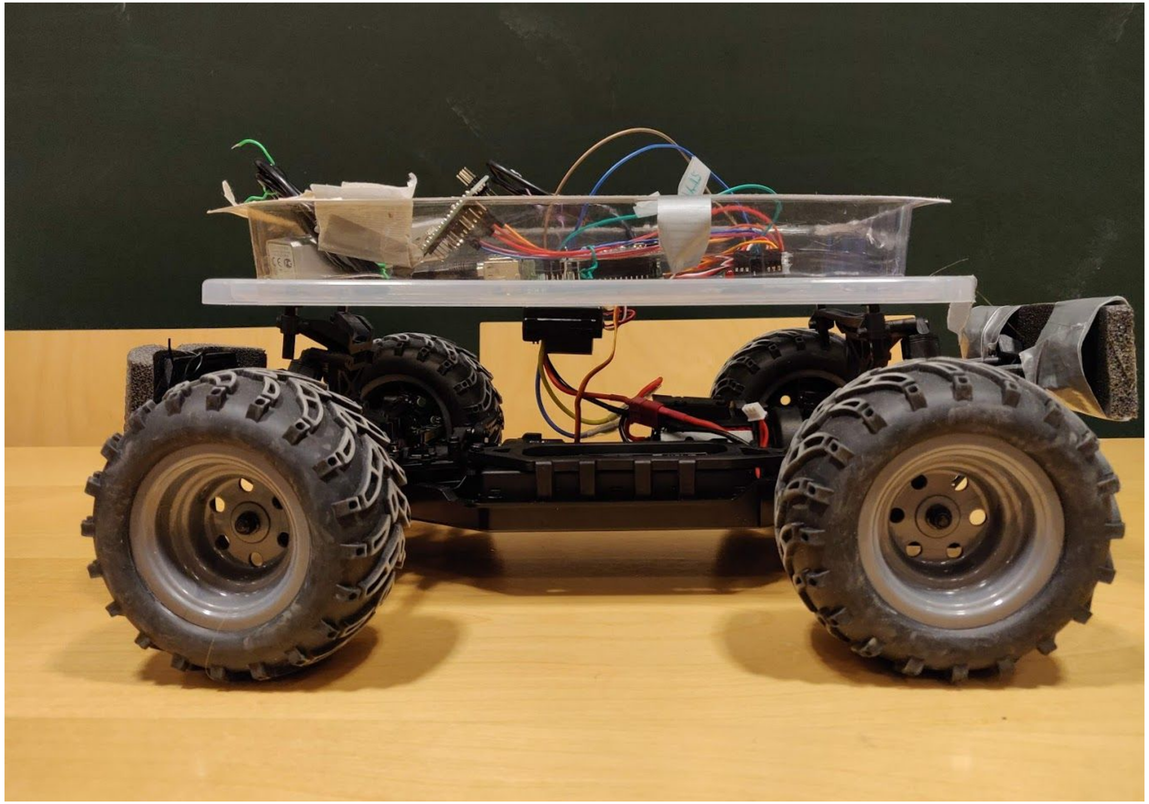
3. Hardware





Hardware connections and data flow

Testing environment



Car from side

4. Important Concepts

What are ultra wideband signals?

- Common definition: An UWB signal has
  - an absolute bandwidth (B) or “-10 dB bandwidth” of at least 500 MHz **OR**
  - a fractional bandwidth (B_{frac}) larger than 20%
- B_f<1%: narrowband, 1% < B_f < 20%: wideband, B_f>20%: ultra-wideband
- Very short duration waveforms

$$B = f_H - f_L$$

$$B_{\text{frac}} = \frac{2(f_H - f_L)}{f_H + f_L}$$

Why use UWB?

- High speed
  - High bandwidth allows high transmission speeds
- Low cost and power
  - Simple hardware like CMOS
  - Signal power can be kept low
- Penetrates obstacles
  - Low frequencies in combination with high frequencies
- Low latency/High accuracy
  - Wide spectrum in frequency domain leads to high resolution in time domain

4a. Kalman Filter

- Estimate our state to reduce the noise
- State-space, sensor data
- Implemented using Python package FilterPy

$$
\begin{aligned}
x_{k+1} &= F_k x_k + B u_k + w \\
z_k &= H_k x_k + v.
\end{aligned}
$$

Kalman Filter for Moving Object

Process model:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix}$$

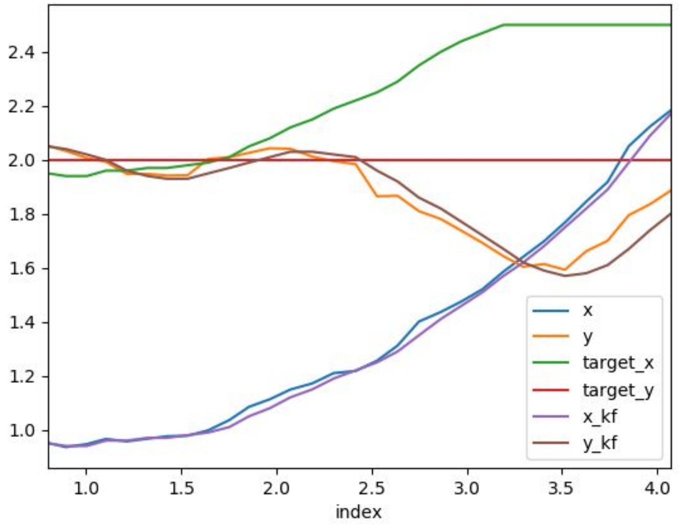
$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \Delta t^2 & 0 \\ 0 & 0 & 0 & \Delta t^2 \end{bmatrix}$$

Measurements: x and y position from UWB

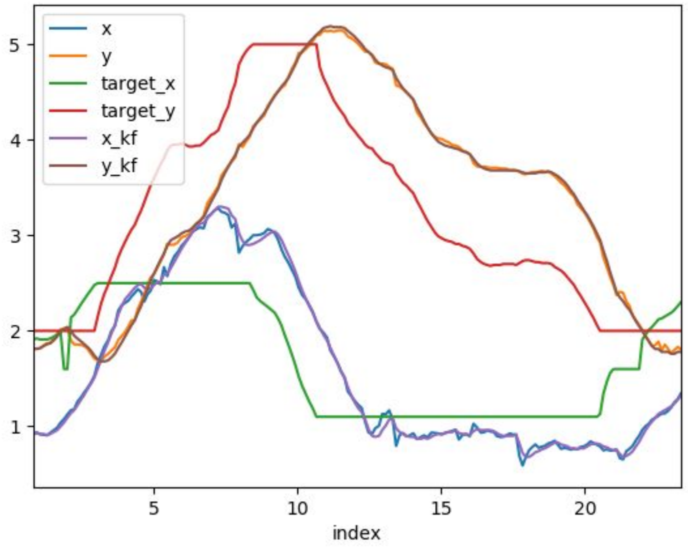
$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

During a linear path:



During a rectangular path:



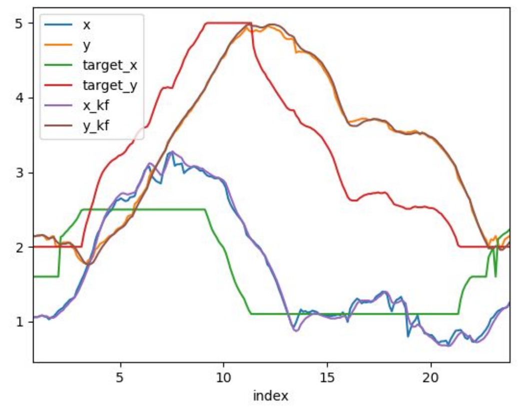
Kalman Filter for Moving Vehicle

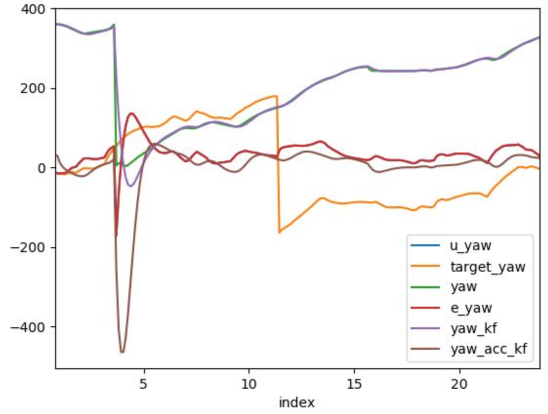
Measurement:

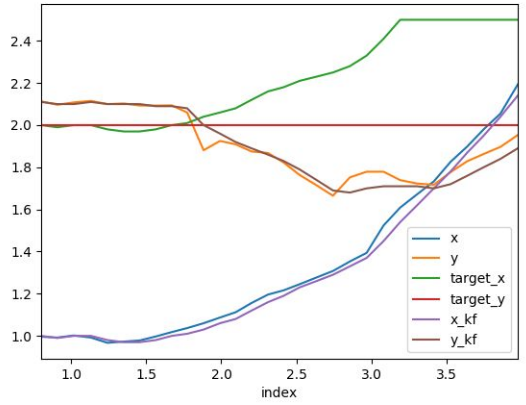
- Position data from UWB
- Heading and acceleration in x and y directions from IMU

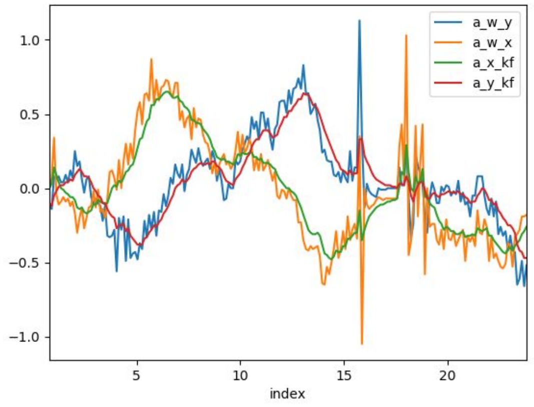
Process model:

$$x = \begin{bmatrix} x \\ y \\ \theta \\ \dot{\theta} \\ \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} \quad F = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ c \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, u = [\phi] \quad z = \begin{bmatrix} x \\ y \\ \theta \\ \ddot{x} \\ \ddot{y} \end{bmatrix}$$









Big Kalman filter results

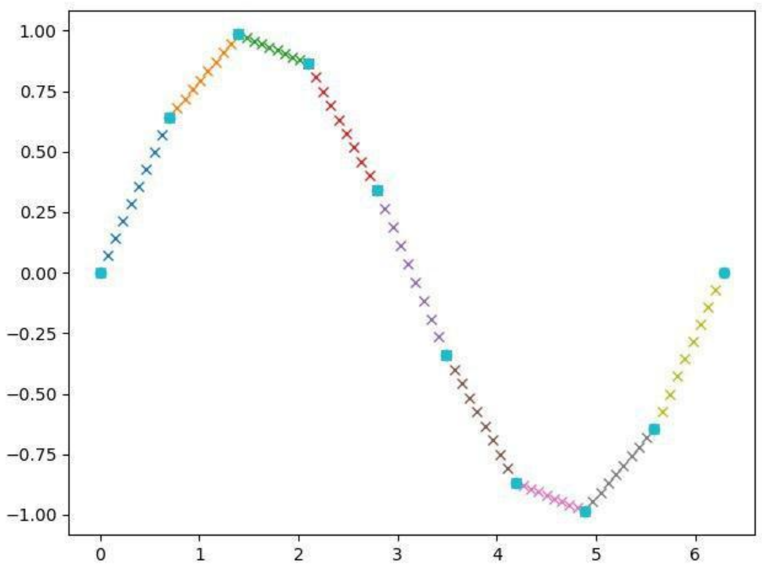
Motion planning and control

- Pathfinding
- Path drawing
- Path tracking

4b. Pathing

- Problem: Interpolate list of destinations to create a continuous path

- Bresenham's line drawing algorithm
- Cubic splines

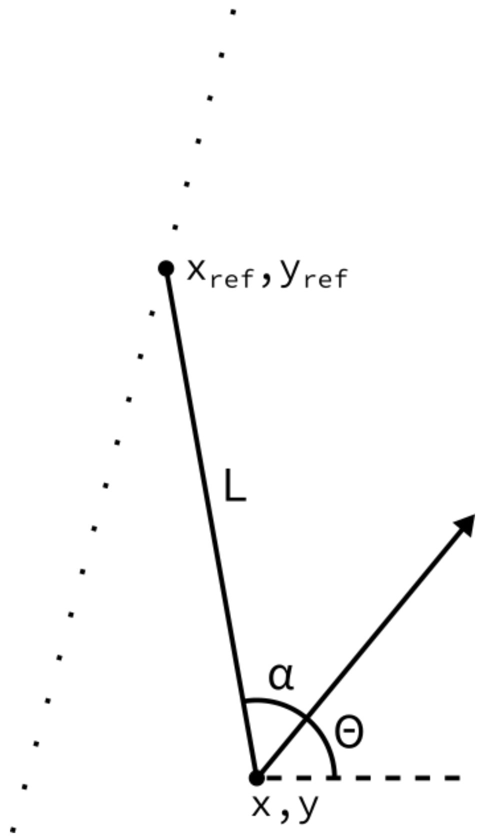


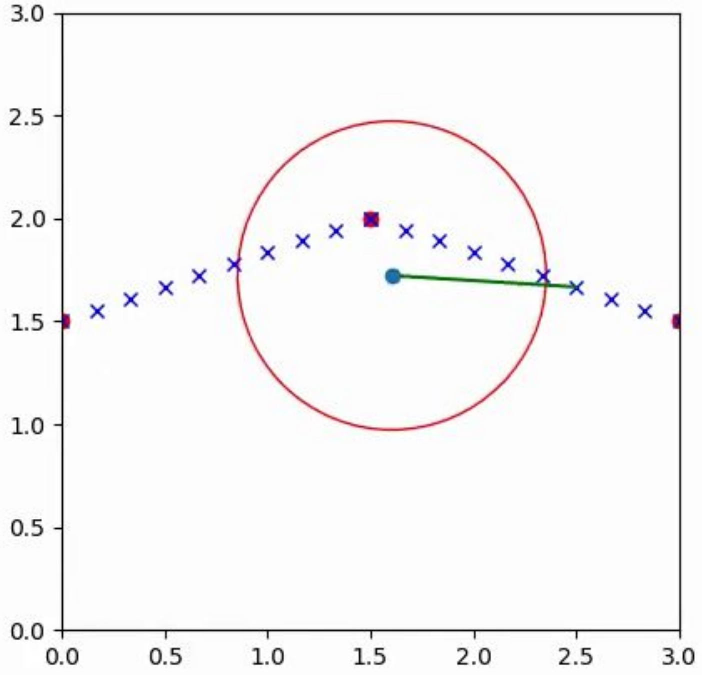
4c. Pure Pursuit

- Path tracking algorithm used since the 90's
- Many variations
- Drive toward a point that is always ahead of the car.

- Find closest point
- Step forward until a point with distance L to the car.
- Find angle of vector that passes through car and point.
- Calculate difference to the heading angle.

$$\alpha = \arctan\left(\frac{y_{\mathrm{ref}} - y}{x_{\mathrm{ref}} - x}\right) - \theta$$





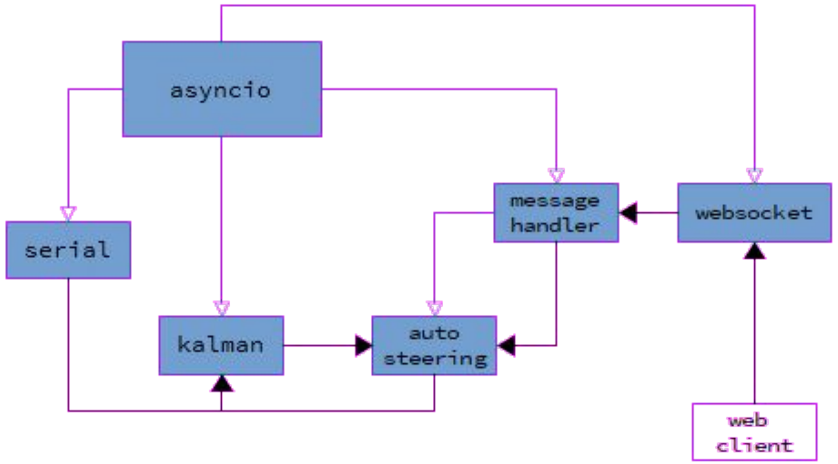
Simulation of Pure Pursuit

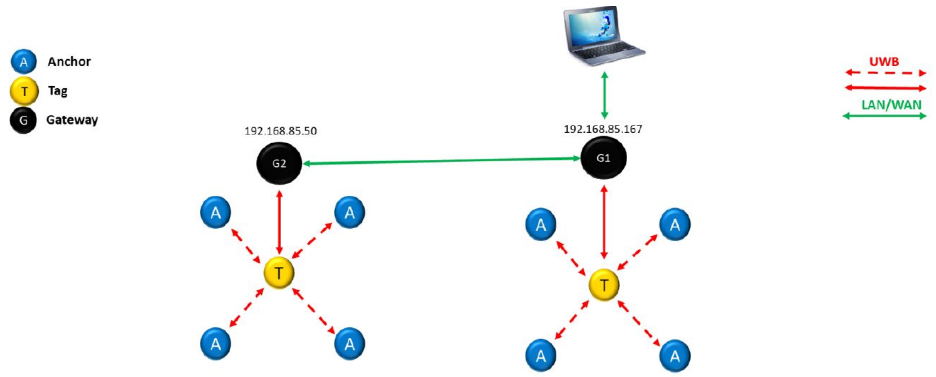
5. Software

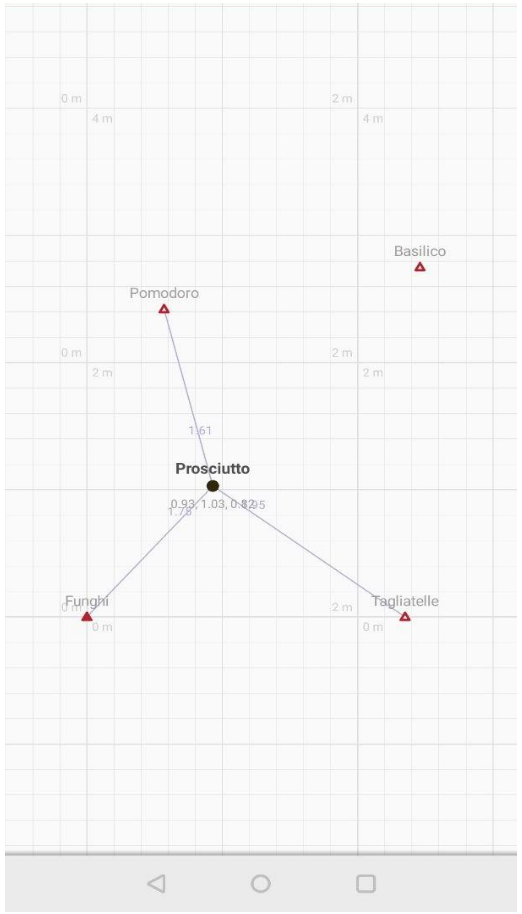
Program Overview

- Python
- Program built around asyncio.
- Tasks running with cooperative concurrency.
- About 9 measurements per second.

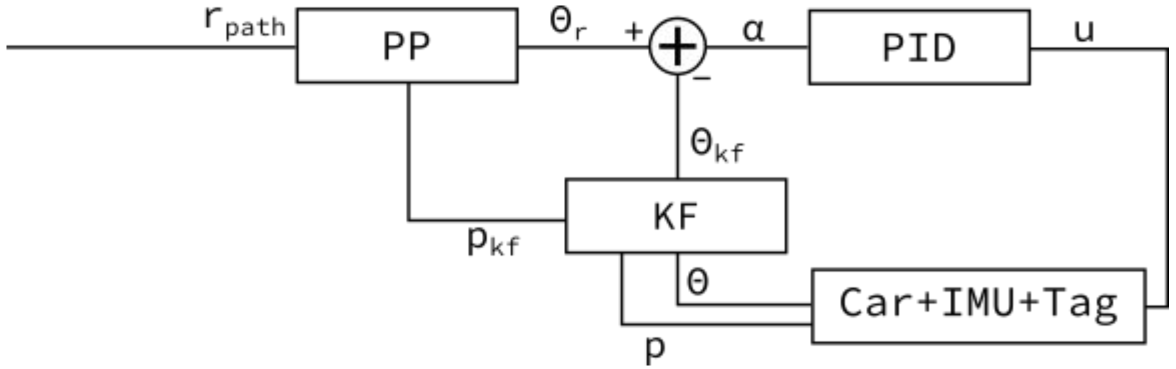
- Arduino software: modified IMU example
- Decawave firmware for positioning



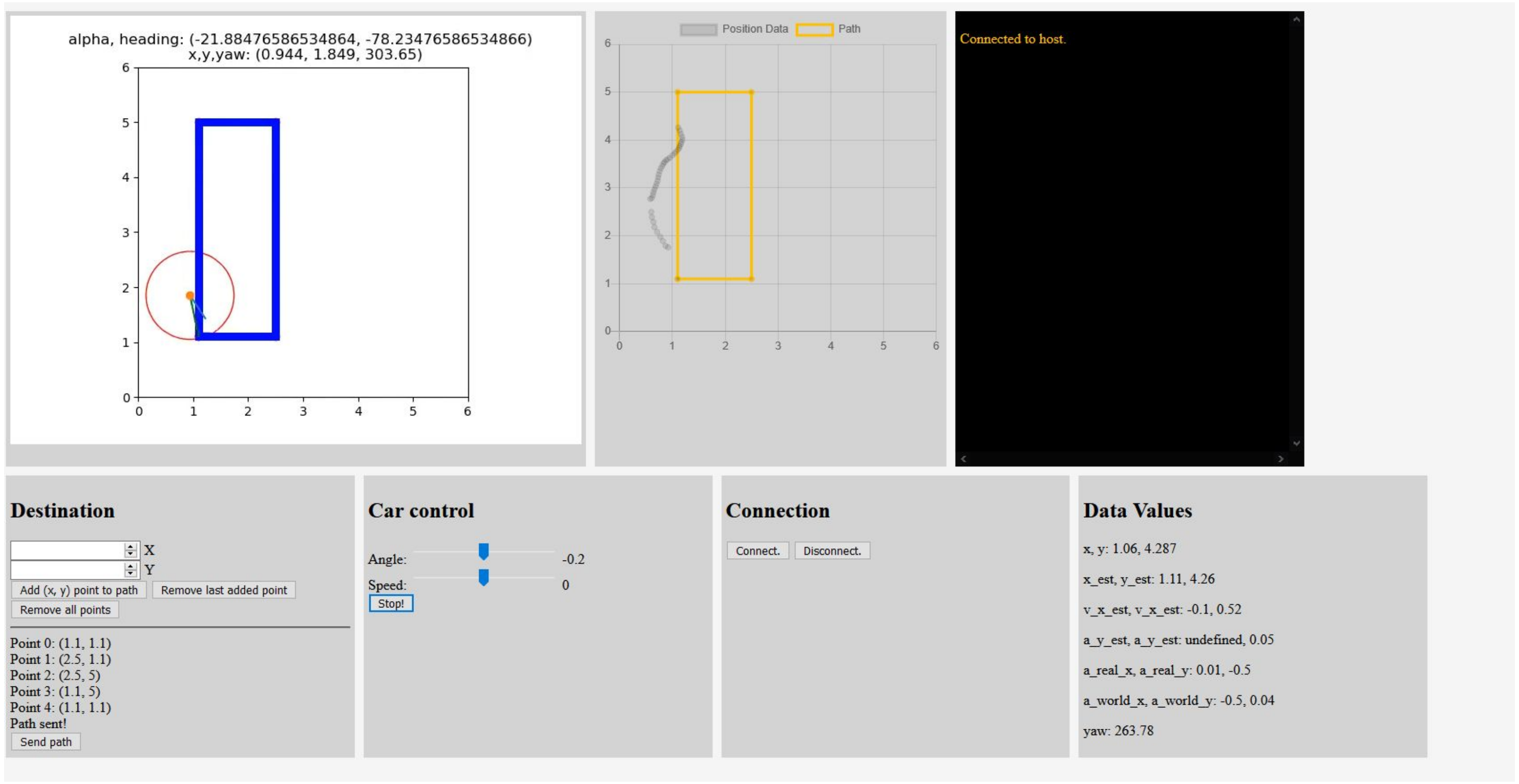

Anchor
Tag
Gateway
192.168.85.50
192.168.85.167
G2
G1
A
T
UWB
LAN/WAN


0 m
2 m
4 m
Basilico
Pomodoro
1.61
Prosciutto
0.93, 1.03, 0.82
1.78
1.95
Funghi
Tagliatelle

MDEK1001



Automatic control system


alpha, heading: (-21.88476586534864, -78.23476586534866)
x,y,yaw: (0.944, 1.849, 303.65)
Position Data
Path

Connected to host.

Destination

X
Y
Add (x, y) point to path | Remove last added point | Remove all points

Point 0: (1.1, 1.1)
Point 1: (2.5, 1.1)
Point 2: (2.5, 5)
Point 3: (1.1, 5)
Point 4: (1.1, 1.1)
Path sent!

Send path

Car control

Angle: -0.2
Speed: 0
Stop!

Connection

Connect. | Disconnect.

Data Values

x, y: 1.06, 4.287

x_est, y_est: 1.11, 4.26

v_x_est, v_x_est: -0.1, 0.52

a_y_est, a_y_est: undefined, 0.05

a_real_x, a_real_y: 0.01, -0.5

a_world_x, a_world_y: -0.5, 0.04

yaw: 263.78

Browser interface

Animation of path tracking

6. Grading Criteria

Grading Criteria pt. 1

Grade 3

- Good knowledge of ultra wideband technology
- Good knowledge of localization technology
- Good knowledge about Kalman filters and application of position estimation techniques
- Good knowledge of automatic control
- Text based UI in browser that show the position values of the anchors and the car/tag.
- The Raspberry Pi, and by extension the car, will be controlled by a simple command line based interface
- The car works along the straight line, and can stop at the assigned endpoint with an accuracy of no more than the car's length

Grading Criteria pt. 2

Grade 4

- Implementation of a PID control system to steer the car with good stability
- A web based UI that shows position data and has manual control of the car, such as [STOP], [START]
- The car works along a curved line, and can stop at the assigned endpoint with an accuracy of no more than the car's length

Grade 5

- The car works along a specified closed path, and can stop at the assigned endpoint (= start point) with an accuracy of no more than the car's length
- The web based UI can visualize the whole path of the car

Reflections

- UWB localisation is fast, energy efficient and not very noisy
- No real time operating system used, Raspberry Pi is powerful.
- Python has been useful but hard to debug.

Future work

- 9 DOF IMU
- Statically typed language (Go, Rust, C++)
- More “flexible” vehicle (depending on problem to focus on)
- Less reliance on Decawave
- Nonlinear Kalman filter
- Better method of parameter optimization