Specs and architecture V0.5

1. Physical

Dimensions intérieures chassis = 53 x 85

H axe roue avant = 9cm

H axe roue arrière = 12cm

1. Architecture HW and key functions
   1. Overview

* Main unit (Pi)
* Traction and position control board (Arduino or Teensy 3.2)
  + Compass
  + GPS
  + Decoders
  + Motor drivers
* Environment sensors board
  + US sensor
  + Others (?)
* Battery control board
  1. Main unit – Pi
* MU1 - Webserver for HMI
* MU2 - Time reference
* MU3 - Maintain status database
* MU4 - Calculate trajectories
* MU5 - Drives Traction and position board (SPI 1)
* MU6 - Drives environment sensors (SPI 2)
  1. Traction and Position control board (Arduino or Teensy 3.2)

SPI-1 slave, I2C-1 master

* TP1 – receives and decodes order from MU (SPI bus)

Order can be a segment (distance, speed), a turn (target heading)

{“type”:”segment/turn”, “distance”:1.0, “speed”:100, / “target\_heading”:180}

* TP2 – calculate PWM x4 to execute order and set them
* TP3 – poll encoders (I2C-1 slave) frequency = 0,1s
* TP4 – poll compass (I2C-1 slave) = 0,1s
* TP5 – poll GPS
* TP6 – adjust PWM to deliver order = 0,1s
* TP7 – feedback actual segment to MU (SPI) = 0,1s
  + 1. Decoders
  + D1 – read (interrupt) encoder ticks

Interrupt overhead = 2,93micros before + 2,18micros after = 5,1125microsec in total + interrupt instructions time (ATmega 328 16Mhz) + 4,78microsec for a digital read or 1microsec for a register read = 7microsec – say x2 for overhead = 14microseconds. Will get 5\*70=350 interrupts per second per encoder => 4,900 microseconds = 5 miliseconds per second (0,5% per encoder). Sounds OK !

@ 350 interrupts

Cycle = 3ms

* + - Interrupt management: 14microsec x4 = 56microsec
    - I2C reading for 4 x2 bytes = 1ms (source: compass test) @ 10hz => 1ms/30 = 30microsec
* CPU load of 86microsec / 3ms = 2,8% = ok
* Leadtimewise, I2C to be executed between external interrupts. Should be fine because CPU is free 98% of the time  
  Need to check if I2C

2 options:

* + - 2 Uno reading 2 encoders each (seems an overkill)
    - Use 1 Mega Board (5V), Zero (3,3V) able to manage 6+ external interrupts. Need to confirm the queeing of interrupt will not delay I2C call for ever + latching is ok
  + D2 – count / decount based on direction

Interrupt on encoderA (raising, falling), check encoderB for level and increment/decrement

* + D3 – provides value since last poll

Time based interrupt (0,1s), latch data

Act as an I2C slave and deliver the data to PT

Issue = difficult to synchronize if two decoders boards - use a change interrupt to cascade Time based interrupt of on board to the other.

* + 1. Compass
  + C1 – provides heading

I2C call from master

* + 1. GPS

tbd

* + 1. Motors drivers
  + MD1 – sets direction

Basic execution of PT sketch

* + MD2 – execute PWM

Basic execution of PT sketch

* 1. Sensors board

tbd

1. Functional specifications
   1. A. Main Unit
      1. MU 4 – Calculate trajectories

2 modes = waypoints and area to be mowed

* + - 1. Way points
  + Load waypoints

Waypoints file structure (tbc)

{{“type”:”waypoints”},{“name”:“toto”},{“status”:”valid/invalid/unknown”}}

{{x,y},

…

{x,y}}

* + Calculate segments
    - Seg0\_origin=starting point
    - Draw straight line to next way point
    - Current\_block = next crossed block
    - Check for forbidden blocks (block crossed + 3x3 around)
    - If ok Segi\_end = current\_block else Segi\_end = last\_block
    - Add waypoint @ 90° left + 3 blocks from last ok block
  + Play segments (MU5)
    - 1. Mowed area mode

Based on and random action within mowed area

* + load mown\_area

Mown\_area file structure

{{“type”:”area”},{“name”:“toto”},{“status”:”valid/invalid/unknown”}}

{{x,y},

…

{x,y}}

* + head towards mowed\_area
    - Find closest point of mowed\_area as waypoint 1
    - Calculate route (waypoint mode)
    - Enter into mowed\_area
  + Mowing trajectory algorithm
    - To be implemented – check for forbidden blocks and redesign mowed area to exclude them
    - Alternative 1
      * Generate random heading with heuristic based on maximum number of unmowed blocks
      * Calculate waypoint as intersection of heading and mowed\_area border
      * Go back 1 block
      * Turn by random heading generated based on heurisitic
    - Alternative 2
      * Generate inner border of mowed area moving all waypoints by one block towards the inner (need to manage narrow areas)
      * Mow along this line
      * Repeat same logic
    1. MU5 – Drive TP board

Send all segments or send one by one (pushed/pulled - pulled is better than all at a time, but still requires to wait for TP board to change course)

Push/Pull (?), two segments initially then ok & n+2 or stop & n’+1 & n+2 at the end of n execution.

Need for TP to report on position => TP shall act as a master on this as part of the regulation loop.

* 1. TP functionalities
     1. Calculate PWM x4 to execute order and set them

If (order[“type”]=”segment”)

Set all PWM=100

How is distance managed?

If (order[“type”]=”segment”)

Set right\_pwm=50 & right\_pwm=-50

* + 1. TP6 – adjust PWM to deliver order
* Execute after TP 4 or TP 5 (Encoder ticks and compass available every 0,1s)
* Straight line management (forward and backward)
  + If max (current gap; abs(sumof(current gap and previous gap))) between slowest left and slowest right >= 2 ticks (6% deviation):
    - slowdown fastest side (function of the gap tbd)
    - Reset current gap #fixed so do not carry forward
  + Else if sumof(current gap and previous gap)=0 :
    - Reset current gap #current and previous cancel out => do not carry forward
  + If gap between front and rear
    - Slowdown slipping wheel to slowest wheel of the corresponding/both side(s)
  + If no gap front rear and abs(sum(current gap – previous gap) <=1:
    - Increase speed by 10% on all
  + Former\_gap=current\_gap
* Distance management - tbd
* Rotation management
  + Pour un chassis de 60 x 90, circonférence cercle circonscrit = 3,4m => vitesse de rotation ~ 52°/seconde => roation à faire mi-vitesse soit 26°/s soit avec une fréquence de 5hz sur le compas une resolution de l’ordre de 5,2° qui est 2x la precision du compas.
  + Check if abs(current\_heading - target\_heading) < 10° then PWM=25%
  + Check if abs(current\_heading - target\_heading) < 2° then PWM=0
  1. Sensors board

1. Architecture

2.1 overall logic

Backend updates/reads from SQL database (emergency stop – use direct action?)

Frontend based on webserver

2.2 front end

Webserver:

\* main HMI page

\* waypoints / mown\_area loading

\* specific waypoints/mown\_area file selection

Main HMI page

Actions

\* mode selection (waypoints, mown\_area, coordinates\_acquisition)

\* Start / pause / resume

\* Emergency stop

\* Go to base

\* Adjust Speed (50-100%)

\* Basic manual remote control for coordinates\_acquisition mode and manual test

Tbc - Stop ICE engine

Info

\* state of ICE engine

\* Battery level

\* Position X,Y (P2 position on a map)

\* Speed

\* Traction power

\* Errors & warnings messages

Waypoints / mown\_area upload

\* select file from local drive

\* upload

\* check structure

\* Confirm, place in relevant directory and make active or raise exception

Waypoints/ active area selection

\* list of available files in each category (2 boxes)

\* make active

2.3 backend

\* SQL database - all variables with time stamp  
BE CAREFULL LIFETIME OF SD CARD

\* main routine

Class Mower

* datas

x, y, speedx, speedy, messages, ice\_status, battery\_level, traction\_power

* functions

get\_info() #Collect position, calculate speed, calculate estimated\_heading, get magnetic\_heading,

store in database

emergency\_stop()

manual\_forward(meters)

manual\_backward(meters)

manual\_turn\_right(degrees)

manual\_turn\_left(degrees)

set\_speed()

Class Routing()

* datas

mode, routing, next\_step

* functions

create\_routing(mode, file)

go\_to\_base()

calculat\_next\_step(mower, routing)

execute(mower, next\_step)

pause()

resume()

Class GPS()

Data

Functions

Initialization

launch webserver

mower=Mower()

while routing is not set :

pass

mower.create\_routing()

main loop

mower.get\_info()

check database for new orders

mower.calculate\_next\_step()

mower.execute()

2.4 Communication protocol

http

Robot connects to wifi

Robot Web server - Webpage of the robot