Steering Control for Autonomous Formula

2016 Model

Alon Spinner, Idan Roth, Ohad Mochly

# Contents

[Contents 2](#_Toc11325380)

[1 Problem Statement and Solution Architecture 3](#_Toc11325381)

[2 Solution One (abandoned) 3](#_Toc11325382)

[2.1 Transmission Angles for First Solution 3](#_Toc11325383)

[3 Solution 2 - 26/05/19 4](#_Toc11325384)

[3.1 Accomplishments up to 26/05/19 4](#_Toc11325385)

[3.2 What needs to be done: 6](#_Toc11325386)

[3.3 Connecting to Arduino through the Jeston 6](#_Toc11325387)

[3.4 Information we used to figure out things 7](#_Toc11325388)

[4 Accomplishments 03/06/19 10](#_Toc11325389)

[4.1 Updated Faulhaber Code: 10](#_Toc11325390)

[4.2 Calibration Protocol 10](#_Toc11325391)

[5 Accomplishments 09/06/19 11](#_Toc11325392)

[5.1 Updated Arduino Code 11](#_Toc11325393)

[5.2 Python Code 11](#_Toc11325394)

# Problem Statement and Solution Architecture

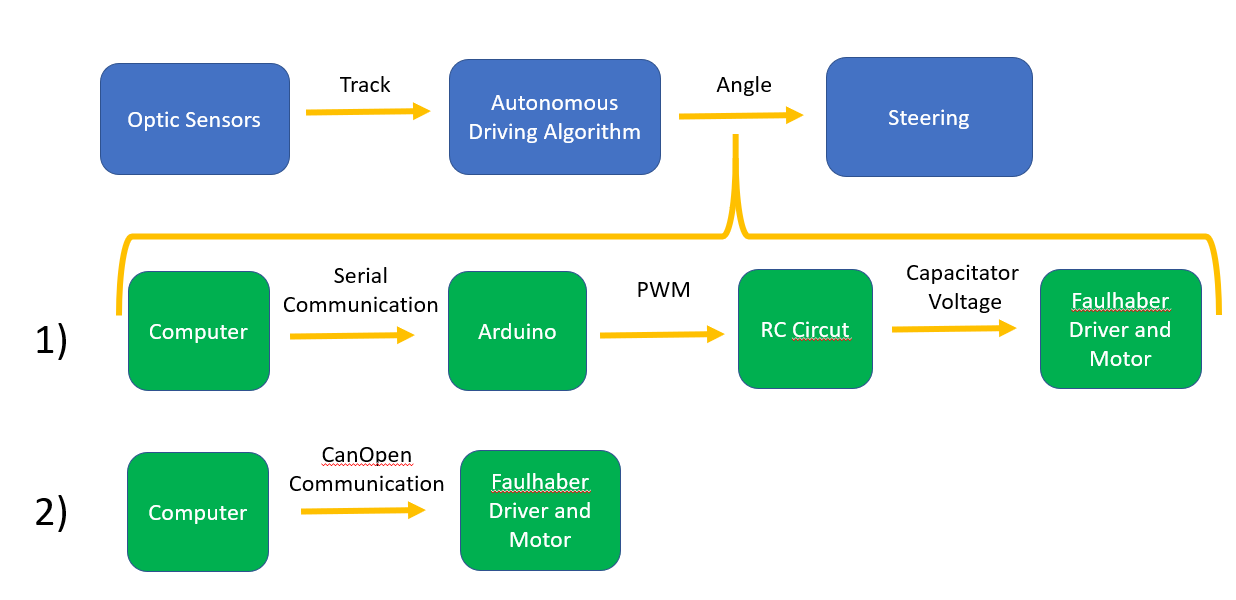
Using a Faulhaber Driver (MC 5010 S) and a Faulhaber Brushless DC Motor (3274 BP4) with Hall Sensors and 1:100 Gear Head, move the steering wheels with an angle given by the autonomous computer algorithm.

# Solution One (abandoned)

Two solution lines were though up:

1. Quicker and dirty serial communication + Arduino for analog input to driver
2. CanOpen (required for competition)

It was decided to run with the first option before attempting the second.



## Transmission Angles for First Solution

Steering angles range were decided to be

* MATLAB/Cpp sending an integer ranges to an Arduino through Serial Communication.
* The Arduino maps and outputs a corresponding PWM signal to an RC circuit.
* Voltage measured on capacitator ranges is sent through analog in channel to Faulhaber driver.
* The Faulhaber obtains the analog signal in ranging from which maps to

Mapping goes as follows

Where:

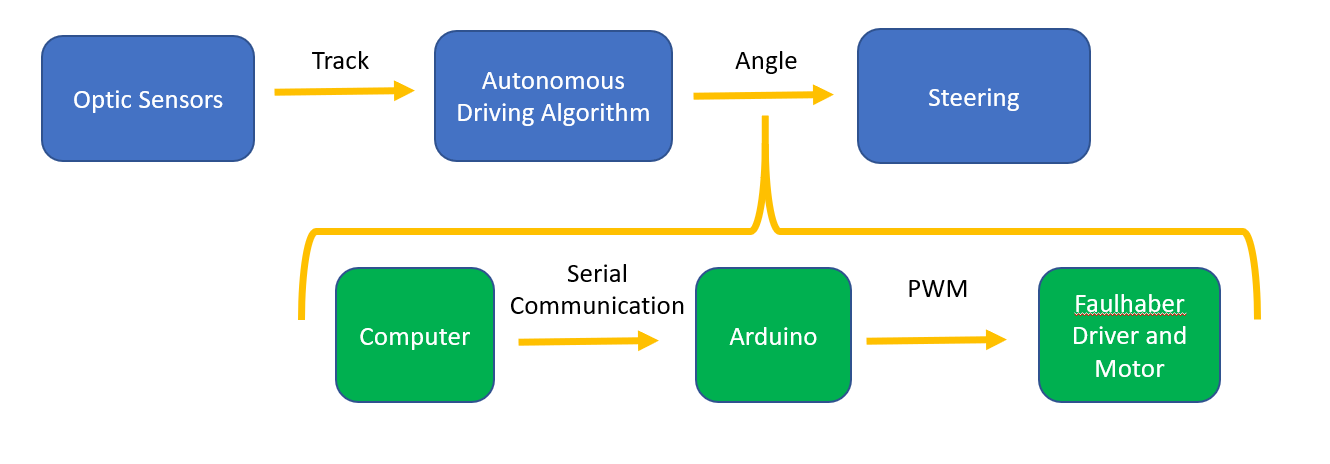
– angles to increments

– motor to pinion

- pinion to wheels ­

# Solution 2 - 26/05/19

## Accomplishments up to 26/05/19

1. Established new architecture for solution  
   
2. Managed to control position live with analog input via the following code

Cable connections – input and outputs – are defined in the Faulahber code in the description.

MATLAB:

%% Send to Arduino

%% initialize port (run once)

s = instrfindall;

if ~isempty(s)

fclose(s);

delete(s);

end

s = serial('COM13','BaudRate',9600,'DataBits',8,'Stopbits',1, 'parity','none');

fopen(s);

%% Load input

Deg=[0,48,0,24,48,12];

%% Transmit Data

T=2; %seconds

for k=1:length(Deg) %U variable name

fwrite(s,Deg(k));

pause(T);

end

Arduino:

FaulHaber:

int pwmpin = 3;

int val;

int dutycycle;

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

pinMode (pwmpin , OUTPUT);

}

void loop() {

// put your main code here, to run repeatedly:

while(Serial.available()==0){ }

val = Serial.read();

dutycycle = map(val,0,48,0,255);

analogWrite(pwmpin,dutycycle);

}

'--------------------------------------------

'Author: Eyal Baruch & Alon Spinner & Amir Saad

'Date: 26/5/19

'--------------------------------------------

'Description:

'given input:

'PWM signal produced from arduino connected TO digital input 1

'arduino ground connected TO analog ground

'produces output:

'moves motor into position corresponding linearly TO PWM signal Duty Cycle

'motor position = Const \* Duty Cycle (Duty Cycle ranges [0,32767]) AND corresponds TO [0,255] PWM in input

'Important:

'All information required is found in "Drive Functions" PDF.

'Search FOR 'Analong

'The PWM signal

'--------------------------------------------

#INCLUDE "MotionParameters.bi"

#INCLUDE "MotionMacros.bi"

'need TO add enable motor functions TO this code

'need TO create a homing calibration without breaking things

SETOBJ $6060.00=-2 'define APC mode

SETOBJ $2331.04=07 'define pwm input through digital

SETOBJ $2317.01=1 'PWM Digital input - pin 1

SETOBJ $2317.04=429490176 'Duty Cycle Gain - needs TO be changed

SETOBJ $2317.05=273000 'incremental offset - 24\*100\*/360\*4096 (we want TO start from -24 degrees) - maybe needs TO be changed

SETOBJ $6080.00=100 'maximal velocity - needs TO be changed

## What needs to be done:

1. Calculate all the transmissions required to obtain the correct output at wheels.

Important Points:

* The value in – “Duty Cycle Gain” - needs to be changed.   
  It is responsible for the conversion between the “Duty Cycle Raw Value” and – “Duty Cycle Scaled Value”. I think “Duty Cycle Scaled Value” is measured in Increments (out of 4096 for full motor cycle).

Note1: “Duty Cycle Scaled Value” corresponds to Target Position when APC mode is defined.  
Note2: “Duty Cycle Scaled Value” does not account for gear  
Note3: “Duty Cycle Raw Value” ranges [0,32767]) and corresponds to [0,255] PWM input linearly.

* Gear ratio is 1:100
* Transmission between pinion and motor (via strip) is
* Transmission between wheel angles and pinion is

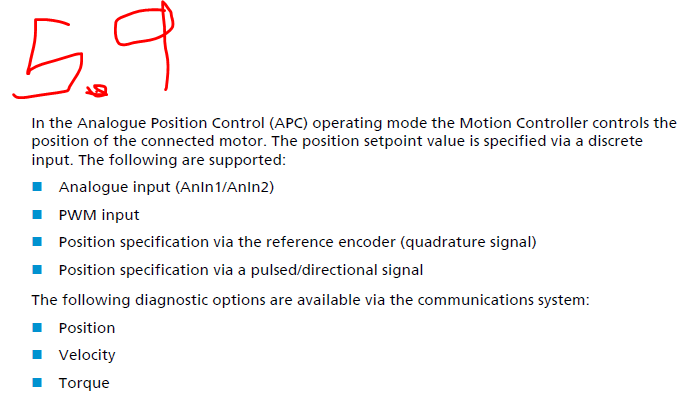
1. Calculate the correct offset so that for input Duty Cycle in PWM the angle will be .
2. Test on wheels when strip is connected!!
3. Add Enable Motor functions (initialization) to the Faulhaber code if you haven’t already.
4. Decide on a maximal speed for the motor to move in and input into Faulhaber code.
5. Find out what is the proper way to do homing (initial calibration of the motor-driver). During homing the motor moves and we can’t allow it to move more than it should as it will brake stuff. Do we need to create a separate code that will be ran when the strip is not connected to the motor’s gear?   
   Perhaps giving boundaries to the position is enough? See
6. Fit the formula vehicle – everything needs a place to sit. Use the 3D printed boxes. Attach to the vehicle’s power supply.  
   Can we use the Jetson? We need one USB port for the Arduino. We have the knowledge of programming the Faulhaber driver and have it operated without a connection to a computer.

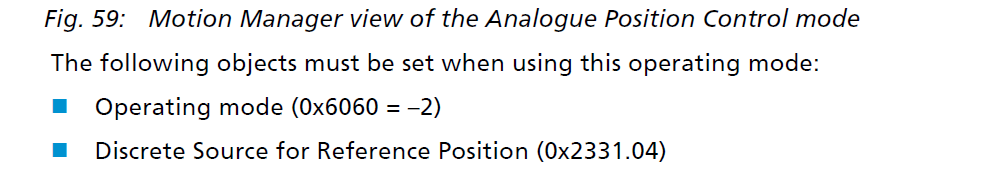
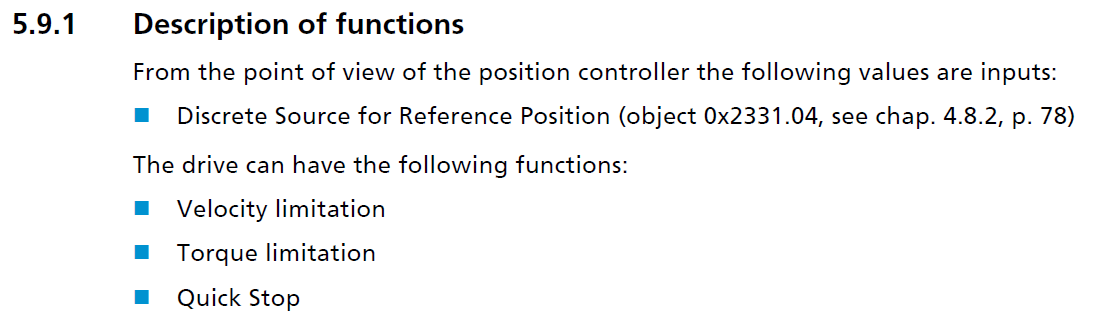
## Connecting to Arduino through the Jeston

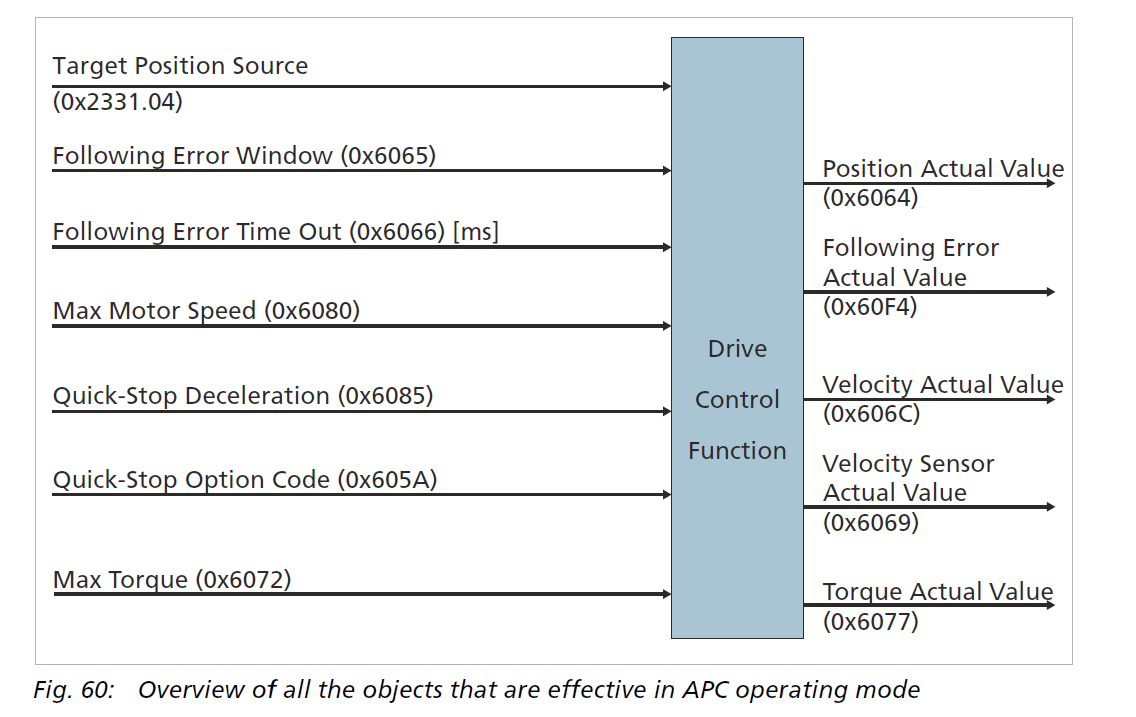
1. Hook up the Jeston properly – it is a task in its own but rather intuitive.
2. Connect Arduino to the only full-size USB port that is on the board.
3. Go the desktop and open the program called “Terminal”
4. Type “cd ~/tmp” to enter the temporary directory
5. Type “Serial <number>” to send values through to the Arduino.  
   <number> = time delay between position values in seconds

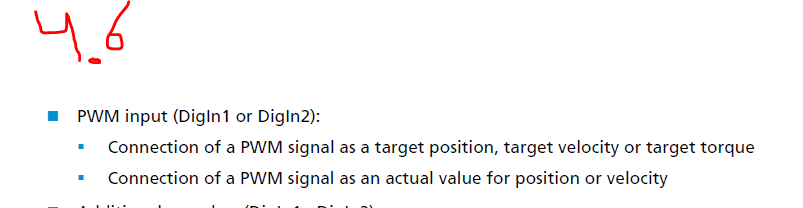
## Information we used to figure out things

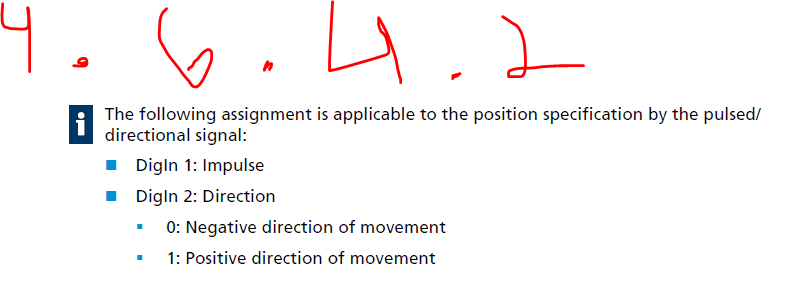
All Information is provided from PDF and object browser.   
Red marking are section numbers in

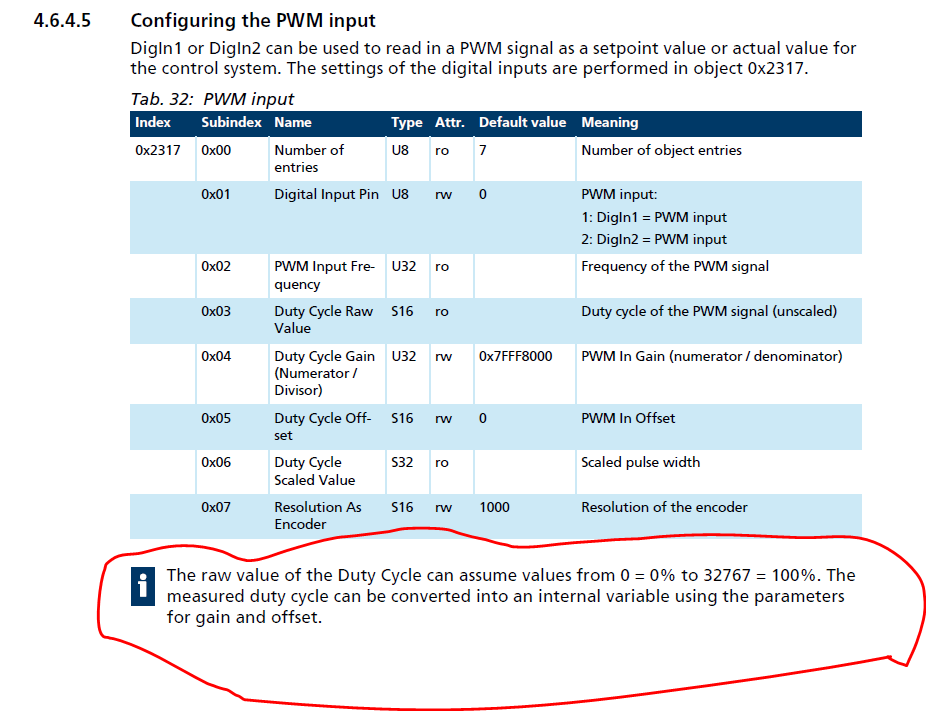


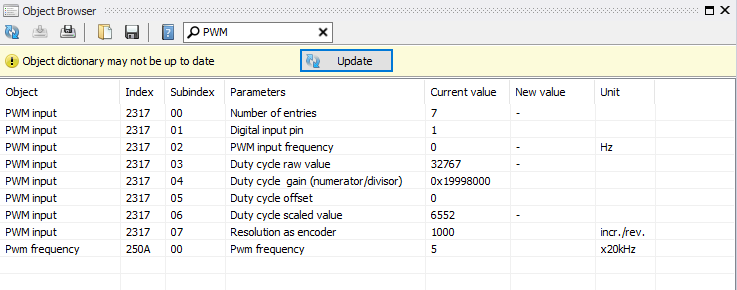












# Accomplishments 03/06/19

## Updated Faulhaber Code:

'Author: Eyal Baruch & Alon Spinner & Amir Saad & Idan Roth & Ohad Mochly

'Date: 26/5/19

'--------------------------------------------

'Description:

'given input:

'PWM signal produced from arduino connected TO digital input 1

'arduino ground connected TO analog ground

'produces output:

'moves motor into position corresponding linearly TO PWM signal Duty Cycle

'motor position = Const \* Duty Cycle (Duty Cycle ranges [0,32767]) AND corresponds TO [0,255] PWM in input

'Important:

'All information required is found in "Drive Functions" PDF.

'Search FOR 'Analong

'The PWM signal

'--------------------------------------------

#INCLUDE "MotionParameters.bi"

#INCLUDE "MotionMacros.bi"

#INCLUDE "MotionFunctions.bi"

MC.SetOpmodeHoming '$6060.00=-2 'define APC Homing

SETOBJ $6098.00=37 'set homing mode method. refernece value is zero.

Enable()

MC.SetOpmodeAPC 'SETOBJ $6060.00=-2 'define APC mode

SETOBJ $2331.04=07 'define pwm input through digital

SETOBJ $2317.01=1 'PWM Digital input - pin 1

SETOBJ $2317.04=$E8B601E0 'Duty Cycle Gain\ Position Factor (2 bytes FOR numinator AND 2 bytes FOR dominator)

SETOBJ $2317.05=49151 'incremental offset - 24\*100\*/360\*4096 (we want TO start from -24 degrees) - maybe needs TO be changed

SETOBJ $6080.00=400 'maximal velocity - needs TO be changed

## Calibration Protocol

1. Disconnect power to Faulhaber
2. Load +24 from range to arduino through Matlab/cpp
3. Move wheels to straight direction position
4. Reconnect Faulhaber
5. Load code onto Faulhaber

# Accomplishments 09/06/19

Moved to Python. Changed script in Arduino to follow

## Updated Arduino Code

int pwmpin = 3; // select the output pin for the PWM

int val; // value [0,48]

int dutycycle; // variable to send the correct pwm value to controller

String str; //string read from python

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

pinMode (pwmpin , OUTPUT);

analogWrite(pwmpin,127); // when restart, send steering to 0 deg

}

void loop() {

// put your main code here, to run repeatedly:

if(Serial.available()>0){

str = Serial.read();

val=str.toInt();

dutycycle = map(val,0,48,0,255);

analogWrite(pwmpin,dutycycle);

}

}

## Python Code

import serial  
ArduinoData=serial.Serial('COM3',9600)  
  
SteeringAngle=24  
ArduinoData.write(chr(SteeringAngle))