Up to date 26/November/2024; 8pm; signed by Sam.

Updated 2/December/2024; 8pm; signed by Sam. Changes are yellow highlighted

Updated 10/December/2024; 11.33mp; signed by Sam. Added FSM flow

NOTE:

1. Vsetky funkcie, ktore mame su v .h hlavickovych suboroch.
2. Dole je vypisane ako ku datumu kod postupuje a co sa vola. Bolo by super z toho spravit vyvojovy diagram ☺. Neslubujem, ze si nato najdem cas..

# File structure:

Include + Source:

1. BASIC
   1. ADC.h/.c
      1. ADC operation, filters for ADC, interrupts for ADC
   2. LED.h/.c
      1. LED control, start-up, braking
   3. Motor.h/.c
      1. Motor control, braking, reversing, PWM control
   4. UART.h/.c
      1. Bluetooth over UART control
      2. Cyclic buffers for sending char, uint and int
      3. Command line with basic commands
   5. main.h/.c
      1. Initializations of peripherals and functions
      2. Main while loop
      3. Command line control
2. ENHANCED
   1. Timers.h/.c
      1. Time-out flags
      2. Time-out variables
      3. Init main clock
      4. Init timer A0, A1, B0
      5. Variable delays
      6. PWM signal generation
      7. Interrupts for time-out features
         1. Brakes LEDs release
         2. Brakes release
   2. Flash\_memory.h/.c
      1. Manage flash memory writes/reads
   3. Data\_temp\_storage.h
      1. Storage of constant values for off-board testing
   4. Data\_operation.h/.c
      1. Correlation algorithms
         1. Auto-correlation
         2. Exact Match
         3. Sum of Absolute Differences
      2. Correlation windows handling
      3. Correlation dumping incorrect samples
      4. Correlation thresholds
      5. Correlation clearing buffers
      6. New-lap-finding functions
      7. Convolution functions
      8. Complex Bluetooth user debug handling
   5. Aux.h/.c
      1. Use saved data for simpler developing
      2. Simple car control
      3. Smart LEDs control
         1. Different axis handling
         2. LED handling
         3. Speed handling
         4. Bluetooth debug for Phyton script
      4. FSM car control
         1. Handling Ref-lap, Run-lap, Stop, Error, Debug, Recovery, Reset states
         2. States transmission
         3. Z-axis handling
         4. LED handling
         5. Map saving
         6. Correlation performing
         7. Proper braking LEDs handling
         8. Complex speed handling
         9. Complex map creating
         10. Recovery and reset
         11. Complex debug possible
         12. Possible training on pre-saved data
         13. Lap counter
         14. Speed translating for code post-processing
         15. Speed adjusting
      5. MAP
         1. Map saving
         2. Map showing
         3. Map creating and segmenting
         4. Map clearing
      6. Complex Bluetooth user debug handling

# Start-up procedure

1. After start up -> main.c = init of all peripherals, infinite loop handling the program
2. Timers are handled in timer.h/.c
3. ADC readings are in ADC.h/.c
4. Bluetooth handlings are in UART.h/.c
5. -> Goes to aux.c = all functions for car controlling. If FSM init -> first REF\_LAP FSM state
6. If calling correlations -> jumps to data\_operation.c = all data operation functions
7. -> aux.c handles positive correlations and proceeds to the next FSM state
8. According to the time-outs and flags proceeds with ADC readings, LEDs controlling.
9. Saves map with every new ADC sample
10. After new lap found, the speed is controlled according to the map and compared to real-time ADC data
11. Runs here until external BLE control or possible stop-conditions are met.

# Screenshots of the Command-line prints

## Start up title

## A screenshot of a computer program Description automatically generatedCommands

## Invalid Command

## A green text on a white background Description automatically generatedEmergency FSM

## A screenshot of a computer screen Description automatically generatedA screenshot of a computer Description automatically generatedMap

## A close-up of a number Description automatically generatedSmart Running State

## A screenshot of a computer Description automatically generatedSegments

## Change of segments

### A screenshot of a computer Description automatically generatedSpeeding up when on a straight segment

### A screenshot of a computer Description automatically generatedChanning speed in a transition zone

## A screenshot of a computer Description automatically generatedEnd of race

# CODE FLOW

Tu skusim popisat podrobne a zrozumitelne ako nas kod funguje ☺

Table of Contents

[File structure: 1](#_Toc184773044)

[Start-up procedure 2](#_Toc184773045)

[Screenshots of the Command-line prints 4](#_Toc184773046)

[Start up title 4](#_Toc184773047)

[Commands 4](#_Toc184773048)

[Invalid Command 4](#_Toc184773049)

[Emergency FSM 4](#_Toc184773050)

[Map 5](#_Toc184773051)

[Smart Running State 5](#_Toc184773052)

[Segments 5](#_Toc184773053)

[Change of segments 6](#_Toc184773054)

[Speeding up when on a straight segment 6](#_Toc184773055)

[Channing speed in a transition zone 7](#_Toc184773056)

[End of race 8](#_Toc184773057)

[CODE FLOW 9](#_Toc184773058)

[Stavy FSM 11](#_Toc184773059)

[STATE\_REF\_LAP 11](#_Toc184773060)

[STATE\_RUNNING 11](#_Toc184773061)

[STATE\_SMART\_RUNNING 11](#_Toc184773062)

[STATE\_STOPPED 12](#_Toc184773063)

[Stavy STATE\_ERROR, STATE\_DEBUG 12](#_Toc184773064)

[Bluetooth Command-line 12](#_Toc184773065)

[Ovladanie LED 12](#_Toc184773066)

[Zatacanie indikovane prednymi LED 12](#_Toc184773067)

[Brzdenie indikovane zadnymi LED 12](#_Toc184773068)

[Ovladania brzdenia 13](#_Toc184773069)

[Ovladanie rychlosti 13](#_Toc184773070)

[Stav nudze 13](#_Toc184773071)

[Algoritmus korelácie 13](#_Toc184773072)

[Posuvne okno 13](#_Toc184773073)

[Vytvaranie mapy 14](#_Toc184773074)

[Segmentacia mapy 14](#_Toc184773075)

[DEBUG 14](#_Toc184773076)

[Indikovanie 14](#_Toc184773077)

[Ulozene ADC data 15](#_Toc184773078)

[Časovace 15](#_Toc184773079)

[Ostatne funkcie 15](#_Toc184773080)

[Komentare v kode 16](#_Toc184773081)

Hlavnou castou kodu je FSM (Finite State Machine), ktora prechadza medzi jednotlivymi stavmi.

Vsetko co sa v kode deje je riadene velmi presnym casovanim, ktore vychadza v timers (casovacov) v MCU.

ADC sampluje cca 20KHz. Nas pouzity filter (moving average), berie 1000 vzoriek z ADC. To znamena ze mame dostupnych 20 vyfiltrovanych vzoriek za jednu sekundu.

## Stavy FSM

### STATE\_REF\_LAP

Toto je defaultny stav po nastartovani auticka. Vsetko v tomto stave sa vykovana v casovych intervaloch 62ms (milisekund).

V tomto stave kod cita ADC vzorky.

Podla ADC vzoriek reaguju predne LED, ktore indikuju prechod pravou/lavou zakrutou.

V tomto stave bezi algoritmus na ukladanie ADC vzoriek do pamate (mapy).

V tomto stave bezi algoritmus na hladanie korelacie.

Ak korelacia nie je najdena po urcitej prejdenej vzialenosti (spocitanej z rychlosti a casu), FSM prechádza do stavu STATE\_RUNNING.

Ak je korelacia najdena (zaciatok noveho kola), spusti sa funcia na vytvorenie segmetov z mapy. Teraz FSM prechadza do stavu STATE\_SMART\_RUNNING.

### STATE\_RUNNING

Ak v stave STATE\_REF\_LAP nie je po urcitej vzdialenosti (nastavene 23m) najdeny zaciatok kola, FSM prejde do tohto stavu.

Tento stav existuje a plny ulohu “zalohy” pre pripad, keby korelacia nie je schopna najst zaciatok kola.

V tomto stave zadne LED plnia ulohu brzd, predne indikuju zakrutu do prislusnej strany.

V tomto stave auticko prechadza zakruty pomalsie a na rovnom useku zrychli. Vsetko je ovladane len z real-time ADC dat.

Iba tento FSM stav sa vykonava v casovych intervaloch 20ms.

Z tohto FSM stavu neexistuje podmienka na prejdene do ineho stavu.

### STATE\_SMART\_RUNNING

Do tohto stavu FSM prejde len ak v stave STATE\_REF\_LAP bola najdena korelacia dat, a teda zatiacok noveho kola.

Kod sa vykovana v casovych intervaloch 62ms.

V tomto stave sa vola funckia, ktora cita mapu. Na zaklade mapy auticko meni rychlost. Ak sa nchadza na rovnom useku, postupne pridava rychlost. V urcitom percente vziadelosti pred prechodom do zakruty (nastavene 25%), auticko podla rychlosti aplikuje brzdenie a spomaluje tak, aby zakrutu bezpecne preslo.

Ak sa nachadza v zakture, pripcip je rovnaky. Pred prechodom do dalsieho segment, rovinky, auticko nastavi rychlost na vacsiu. Akonahle sa nachadza na celom rovnom useku, znova postupne pridava.

V momene, kedy prejde celu mapu, funkcia indikuje jej dokoncenie. Po urcitom pocte prejdenych kol (nastavene 10), FSM prechadza do stavu STATE\_STOPPED.

### STATE\_STOPPED

Do tohto stavu sa FSM dostane len v pripade, ak auticko uspesne preslo 10 kol podla ulozenej mapy.

V tomto stave auticko zastavi.

Predne aj zadne LED indikuju zastavene periodickým blikanim (nastavene 800ms pre toggle).

### Stavy STATE\_ERROR, STATE\_DEBUG

Tieto stavy nemaju presne definovane podmienky a boli vo velkej miere pouzivane pocas debuggovania firmwaru.

# Bluetooth Command-line

Command-line ovladanie bolo primarnym nastrojom ovladania auticka pri vyvoji firmwaru.

V tomto case su implementovane tieto prikazy:

* "help" – vypisanie prikazov
* „start“ – spustenie FSM
* "stop" – zastavanie FSM
* "reset" – restartovanie FSM
* "map" – vypisanie mapy
* "segments" – vypisanie segmentov
* "about" – zobrazenie „o projekte“.

Kod obsahuje jednoduche ISR (Interrupt Service Routine) funkcie, ktore sa staraju o posielanie dat cez UART do BT modulu. Tieto funkcie su jednoduche cyklicke buffre, ktore posielaju postupne po jednom znaku.

# Ovladanie LED

Start auticka je indikovany rychlim 3krat bliknutim vsetkych LED. Toto. Po tomto je algoritmus prechadza do hlavnej while() slucky, kde postupuje podla BL Command-line, alebo automaticky prechadza do riadenia FSM.

## Zatacanie indikovane prednymi LED

Predne LED reaguju na real-time ADC data. FSM stav STATE\_SMART\_RUNNING pouziva samostatnu funkciu na riadenie LED. Tato funkcia navyse obsahuje flags na presne riadenie brzdenia.

## Brzdenie indikovane zadnymi LED

Zadne LED indikuju brzdenie auticka. Akonahle auticko zmeni rychlost na nizsiu, aplikuju sa brzdy a rozsvietia sa zadne LED. Zhasnutie LED a uvolenenie brdz je ovladanie presnym casovacom.

Pri prechode cez zakrutu, ktora je nasledovana dalsiu zakrutou, sa rychlost nezmeni, a teda LED neindikuju ziadne brzdenie. Toto je mozne vdaka vyuzitia casovacov s presnym casovanim a ISR.

# Ovladania brzdenia

Auticko aplikuje brzdy na tak dlhu dobu, akou velkou rychlostou sa pohyhuje, a tak, aby bezpecne preslo po danom useku.

Na presne riadenie je pouzity presny casovac s ISR. Ak je brzdenie v procese, ziadna funkcia nema povolenie zmenit nastavit rychlost auticka. Ked ISR indikuje uvolnenie brzd, FSM je schopna nastavit rychlost auticka.

Volanim funkcie **motor\_brake()** s prislusnou silou brzdenia, brzdi auticko.

# Ovladanie rychlosti

Na ovladanie motora je pouzity casovac timerA, ktory generuje PWM signal.

Tento signal je 20KHz.

Volanim funkcie **motor\_pwm()** s prislusnou rychlostou sa meni duty-cycle casovaca a tym rychlost motora.

FSM ovlada rychlost podla stavov.

Volanie funkcie **motor\_pwm()** moze byt zastavene, je auticko v stave brzdenia.

# Stav nudze

V momente kedy nastava nespravna synchronizacia auticka na realnej trati a citanie ulozenej mapy, auticko reaguje volanim funkci, ktore predchadzaju zrychlovaniu, ktore je indikovane mapou. Realne data v ADC v tomto pripade indikuju nebezpecie (prilis velku rychlost prechodom nebezpecnym usekom (zakrutou)). Auticko silno nudzovo bribrzdi a nastavuje rychlost na taku, ktora je bezpecna na prejdene danym usekom.

Takyto stav moze nastat iba vo FSM stave STATE\_SMART\_RUNNING.

# Algoritmus korelácie

Po intenzivnych testoch kod pouziva vyhradne korelaciu typu Sum of Absolute Differences (SAD). Tato bola schopna vo velkom percente testovanych dat najst zaciatok daneho kola s najvacsiu presnostou.

## Posuvne okno

ADC vzorky su ako prve ukladanie do trv. „referenceho“ pola. Po naplneni tohoho pola sa naplna tzv. „sucasneho“ pole. Po naplneni tohoto pola sa vykona korelacia dat „referenceho“ a „sucasneho“ pola. Tato korelacia je spravila velmi nizkej hodnoty indikujucej nenajdeny zatiatok kola.

S kazdym novym vzorkou ADC sa tato nova hodnota zapise na najnovsiu poziciu pola „sucasne pole“. Subezne s tym sa vymaze hodnota na najstarsom mieste daleho pola a vseky vzorky sa posunu, aby ma pole stale rovnaku velkost.

V tomto momente sa vykovana korelacia.   
Korelacia sa vykovana s kazdym novym vzorkom ADC.

Z podstaty algoritmu SAD, hodnota blizka nule (0) indikuje velmi vysoku korelaciu, tj. zaciatok noveho kola. Po spravnom nastaveni tento minimalnej hranice vystupuje algormitus so zapisanou pozitivnou korelaciu.

Ak je splneni predom nastaveny pocet pozitivnych korelacii (nastavene 1), kod reaguje accordingly.

# Vytvaranie mapy

Mapa je vytorenia z kazdej novej ADC vzorky.

Kod ma moznost nastavenia zahodania prvych nameranych vzoriek (nastavene 2).

Kazda vzorka mapy je oznamena:

* Indexom
* Vzdialenostou prejdenou od poslednej vzorky, resp. nastartovania auticka
* Casom od spustenia auticka

## Segmentacia mapy

Ak bol najdeny zaciatok noveho kola, pred prechodom do noveho stavu sa vykona segmentacia mapy.

Segmentacia znaci rozdelenie mapy na dane useky. Podla tychto usekov bude FSM reagovat zmenou rychlosti auticka, a pod.

Kazdy segment mapy je oznaceny:

* Index segmentu
* Typ segmentu – 0 znaci rovny usek, 1 znaci zakrutu
* Dlzkou useku
* Casom potrebym na prejdeme useku referencou rychlostou
* Vzdianelostou od startu auticka.

Po segmentaci FSM vie aku velku vzdialenost musi auticko prejst a ako dlho to bude trvat prejst referencou rychlostou.

# DEBUG

Debuggovanie algoritmu bolo z velkej casti mozne vdaka prepracovanej Command-line a *ERROR* a *DBG* spravam posielanym cez BLE. Kod obsahuje velke mnozstvo „*ifdef*“ podmienok, ktore umoznuju posielaju tychlo diagnostickych informaci uzivatelovi.

# Indikovanie

Subor *main.c* obsahuje cast kodu, ktory pri starte auticka indikuje DBG mod poslanim spravy cez BLE. Rovnako je mozne odkomentovat prikaz, ktory periodicky blika prednou lavou LED periodou 120ms. Tymto sa indikuje Bluetooth Debug mod.

## Ulozene ADC data

Debug je mozne vyuzivat na citanie predom ulozenych ADC vzoriek zo zavodnej trate. Tieto data su priamo pouzite v algoritme, a tym simuluju testovanie na trati.

# Časovace

Casovace su hojne pouzivane.

Casovat timerA0 je pouzity na generovanie PWM signalu.

Casovace timerA1 a timerB su pouzity na generovanie a ovladanie ISR.

Existuju rozne flagy na ovladanie neblokujuceho cakania.

Existuje funkcia na zadanie neblokujuceho cakania od 1ms. Tato funkcia je **variable\_delay\_ms()**.

# Ostatne funkcie

Kod obsahuje velke mnozstvo funkcii a moznosti, ktore su nevyhnute pre spravne fungovanie, predavanie hodnot, citanie spravnych hodnot, mazanie poli a premennych, a podobne.

Kod rovnako obsahuje velke mnozstvo konstant ***„#define“***, ktore mozu byt zmenene, v .h hlavickovych suboroch.

Niektore zo spomenutych funkci su:

* **car\_control\_simple()**
* **state\_machine\_init()**
* **state\_transition()**
* **state\_machine\_reset()**
* **feed\_stored\_data()**
* **lap\_counter()**
* **show\_map()**
* **dump\_map()**
* **show\_map\_segments()**
* **get\_current\_segment()**
* **get\_speed\_mps\_10()**
* **smart\_car\_leds()**
* **initClockTo16MHz()**
* **init\_timerA0()**
* **init\_timerA1()**
* **init\_timerB0()**
* **corrClearBuffers()**
* **ADC\_init()**
* **ADC\_start()**
* **ADC\_stop()**
* **ADC\_get\_result()**
* **moving\_average()**
* **frame\_samples()**
* **stored\_track\_data\_1()**
* **flash\_write\_data()**
* **flash\_read\_data()**
* **test\_flash\_read\_write()**
* **led\_init()**
* **led\_flash\_all()**
* **led\_brake()**
* **motor\_init()**
* **motor\_forward()**
* **motor\_reverse()**
* **motor\_idle()**
* **UART\_init()**
* **ble\_send()**
* **ble\_send\_uint16()**
* **ble\_send\_int16()**
* **ble\_send\_int32()**

# Komentare v kode

Kod je pomerne bohato komentovany. Kazdy funkcia ma aspon zakladne vyslenie a obsahuje nazvy premennych, ktore indikuju ich spravne pouzitie.