MTD2A: Mobil Train Detection and Action – arduino libray

MTD2A: Model **T**rain **D**etection **A**nd **A**ction – arduino library https://github.com/MTD2A/MTD2A
Jørgen Bo Madsen / V1.2 / 06-06-2025

MTD2A er en samling af avancerede og funktionelle C++ klasser - byggeklodser - til tidsstyret håntering af input og output. Biblioteket er tiltænkt Arduino interesserede uden større programmeringserfaring, der har elektronikstyring og -automatisering som interesse, samt modeltog som hobby.

Fælles for alle byggeklodser:

- Uunderstøtter en bred vifte af inputsensorer og outputenheder
- Er enkle at bruge til at bygge komplekse løsninger med få kommandoer
- Fungere Ikke-blokerende, procesorienteret og tilstandsdrevet
- Tilbyder omfattende kontrol- og fejlfindingsinformation
- Grundigt dokumenterede med eksempler

Bibliotek

```
#include <MTD2A.h>
using namespace MTD2A_const;
```

Nuværende byggeklodser

- MTD2A_binary_input.h
- MTD2A_binary_output.h

Yderligere planlagte byggeklodser

• MTD2A_delay	• MTD2A_stepper
• MTD2A_astable	• MTD2A_display
• MTD2A_flip_flop	• MTD2A_ultrasonic
• MTD2A_tone	• MTD2A_laser
• MTD2A_sound	• MTD2A_IR_ranging
• MTD2A_servo	• MTD2A_DCC_input

MTD2A_const.h

Indeholde brugervenlig globale konstanter til anvendese som parametre til de forskellige klasser og funktioner.

ENABLE	DISABLE
ACTIVE	COMPLETE
FIRST_TRIGGER	LAST_TRIGGER
TIME_DELAY	MONO_STABLE
NORMAL	INVERTED
PULSE	FIXED
BINARY	PWM
RESTART_TIMER	STOP_TIMER
DELAY1MS	DELAY_10MS

Proces faser tilhørende de nekelet moduler

RESET_PHASE = 0
BEGIN_PHASE = 1, OUTPUT_PHASE = 2, END_PHASE = 3
FIRST_TIME_PHASE = 1, LAST_TIME_PHASE = 2, BLOCKING_PHASE = 3
COMPLETE_PHASE = 4

MTD2A_base (h og cpp)

Indeholder styringsfunktioner, fælles interne funktioner og globale fælles funktioner.

MTD2A_globalDebugPrint (); Aktiver fejl og status medddelelser til Arduiono IDE Serial Monitor.
Parameter = ({ENABLE | DISABLE}) Udelades parameter sættes default = ENABLE

MTD2A_loop_execute (); Opdaterer alle instantierede objekte (Linked list of function pointers)
Absolut sidste kommando i void loop (); og skal altid kaldes en - og kun - en gang.

MTD2A_delayTimeMS (); Sætter kadancen hvormed alle instantierede (aktiverede) objekter opdtateres.
Parameter = ({ DELAY_10MS | DELAY_1MS}) Udelades parameter sættes default DELAY_10MS

Kadance og synkronisering

For at sikre at sikre at alle instantierede (aktiverede) objekter ”går i takt”, beregner funktionen MTD2A_loop_execute (); gennemløbstiden af alle instantierede opbjekter samt den kode som brugeren har tilføjet. Funktionen modregner op til 10 millisekunder pr. samlet gennemløb, ved standard kadance = DELAY_10MS

Hvis beregningstiden går ud over de 10 millisekunder kan der opstår problemer at synkroniseringen. Jo længere beregningstiden går ud over de 10 millisekuner, jo flere objekter vil komme ud af takt, og det kan opstå asynkrone problemer. Især ved tidskritiske og hurtigt eksekverende funktioner. I praksis kan det i mange tilfælde fungere, med mindre varietioner, ved en beregningstid på op til 100 millisekunder.

Som udgangspunkt er de mindre kraftige Arduino boards kraftige Fx MEGA,UNO og NANO tilstrækkelige. Hvis der benyttes mange instatierede objekter, anbefales at benytte de mere kraftige Arduino boards (MCU) som fx ESP32 serien.

Beregningstiden kan stige betydeligt ved benyttelse af især flere sensore med egen MCU (fx VL53L4CD laser afstandsmåler), kommunikationsprotokoller fx UART og IIC samt skrivning til Artuino IDE Serial Monitor.
Fx tager det 7-8 miliisekunder at forspørge VL53L4CD om data er klar, og efterfølgende aflæse data.
Kommunikationsprotokoller bør, om muligt, konfigureres til høj hastighed. Det samme gælderfor Serial Monitor, der som stanadrd benytter den langsomme 9600 BAUD hastighed.

Eksekveringshastighed

Hvert faseskift medfører et ekstra gennemløb på 1 - 10 millisekuner. Skal tidstagningen være mere præcis, skal det modregnes i de tider der angives når objektet oprettes.

Eksempel
Der ønskes et præcist samlet tidsforbrug på i alt 1.000 millisekunder.
MTD2A_binary_output green_LED (”Green LED”, 490, 490); 490 + 10 + 490 + 10 = 1.000 millisekunder.

Hvis der er behov for meget hurtig aflæsning og eksekvering, kanDELAY_1MS vælges. Fx hurtige aflæsning infrarød sensore til nødstop. Ulempen er så, at der ikke foretages modregning. Her anbefales det at benytte de mere kraftige Arduino boards (MCU) som fx ESP32 serien.

Eksempel på parallel processering

```
 // Two blinking LEDs. One with symmetric interval and another with asymetric interval.
// Jørgen Bo Madsen / may 2025 / https://github.com/jebmdk

#include <MTD2A.h>
using namespace MTD2A_const;

MTD2A_binary_output red_LED   ("Red LED",   400, 400);
// 0.4 sec light, 0.4 sec no light
MTD2A_binary_output green_LED ("Green LED", 300, 700, 0, PWM, 96);
// 0.3 sec light, 0.7 sec no light, PWM dimmed

void setup() {
  Serial.begin(9600);

  red_LED.initialize   (9);   // Output pin 9
  green_LED.initialize (10);  // Output pin 10

  Serial.println("Two LED blink");
}

void loop() {
  if (red_LED.get_processState()   == COMPLETE)   red_LED.activate();
  if (green_LED.get_processState() == COMPLETE) green_LED.activate();

  MTD2A_loop_execute();
} // Two blinking LEDs. One with symmetric interval and another with asymetric interval.
```