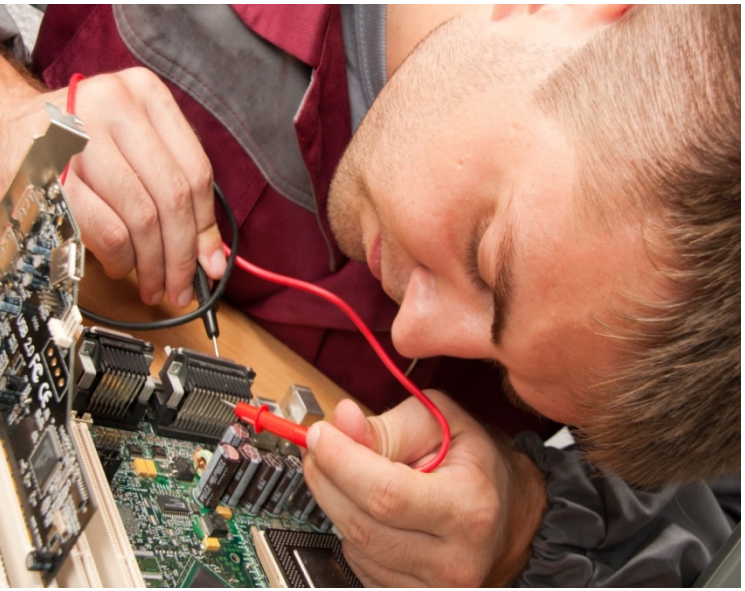
TECHNISCHE HOCHSCHULE DEGGENDORF



NXP Cup Car Implementation

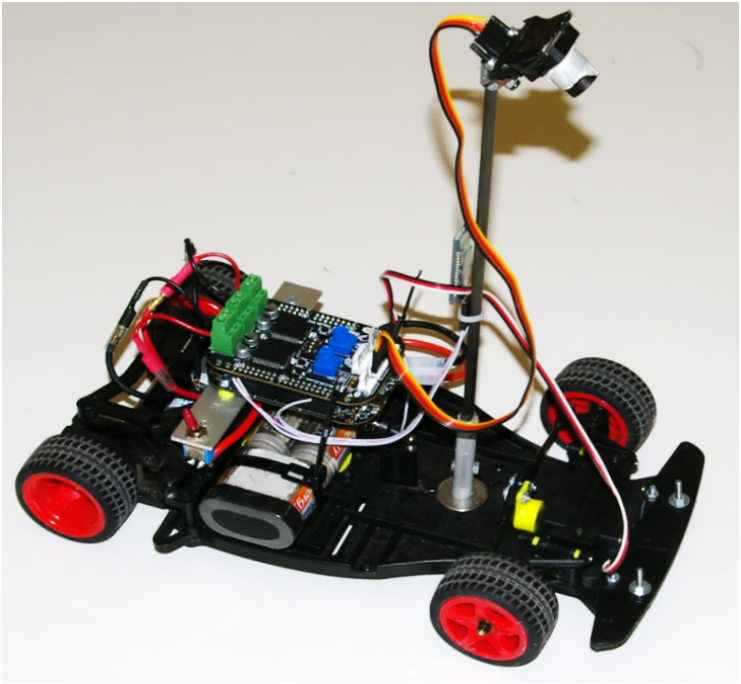
Prof. Dr. Gerald Kupris

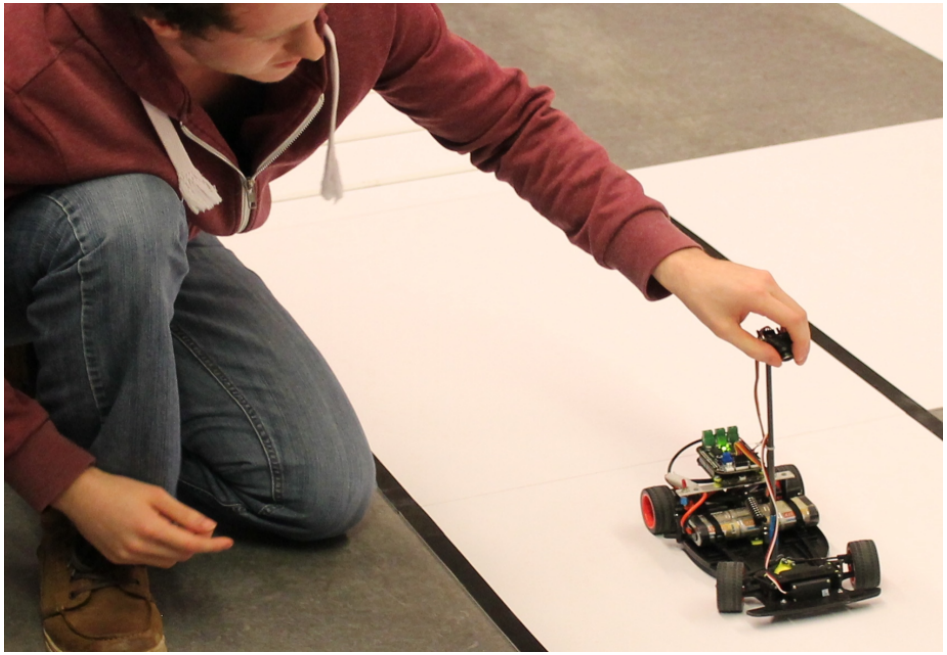
Deggendorf Institute of Technology 16.08.2018

What is NXP Cup?

NXP Cup is a student‘s challenge formerly known as **The Freescale Cup** (TFC). Small model race cars are build by teams of students (only students). The teams of different universities compete with each other on various levels.

All details of the competition are described in the official rule book.

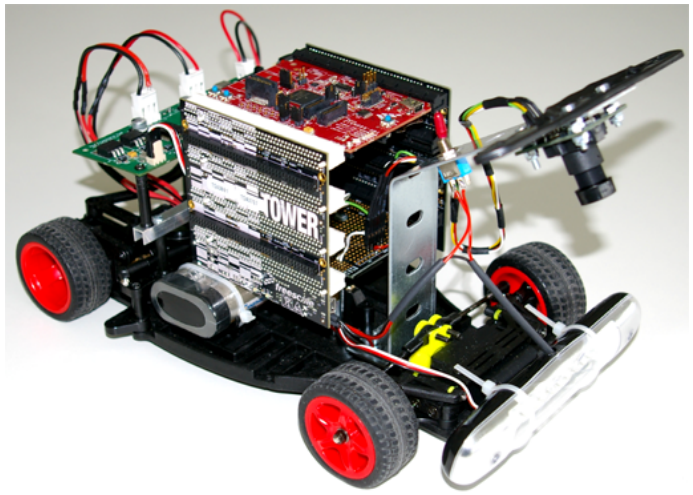




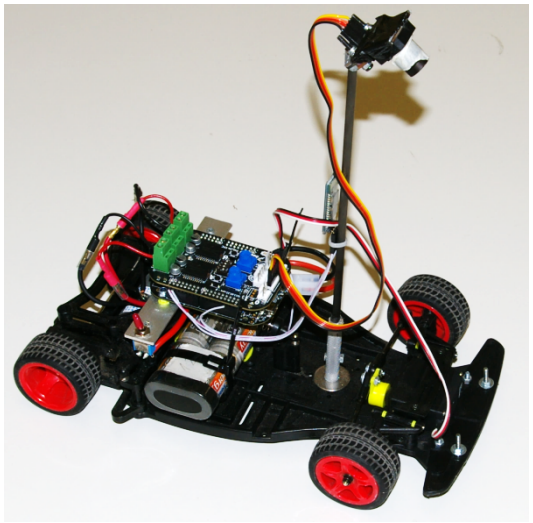
The NXP Cup Car

The car is build from pre-fabricated, standardized parts and is controlled by a microcontroller. All cars have two DC motors for driving, one servo motor and a camera. It should focus on the road and sense the black lines on the road.

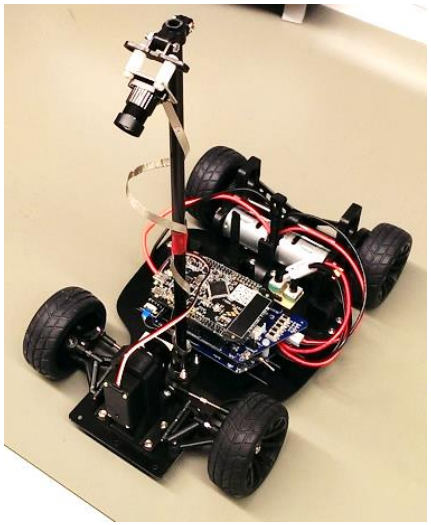
All details of the competition are described in the official rule book.



old model since 2012
(outdated)

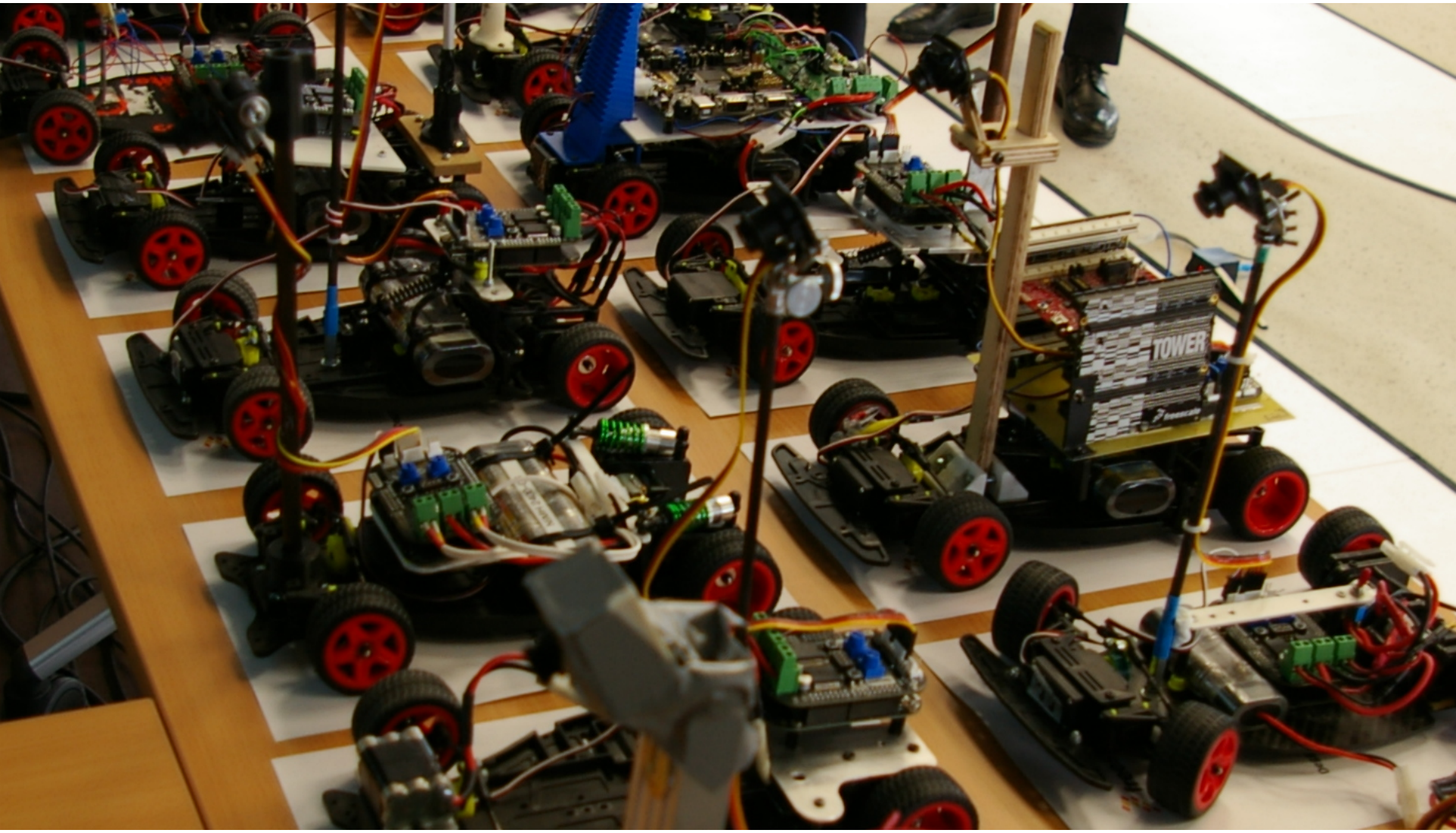


„Model C“ car
(not fabricated any more,
but still in use)



„Alamak“ car
(latest version)

Examples of different NXP Cup Cars



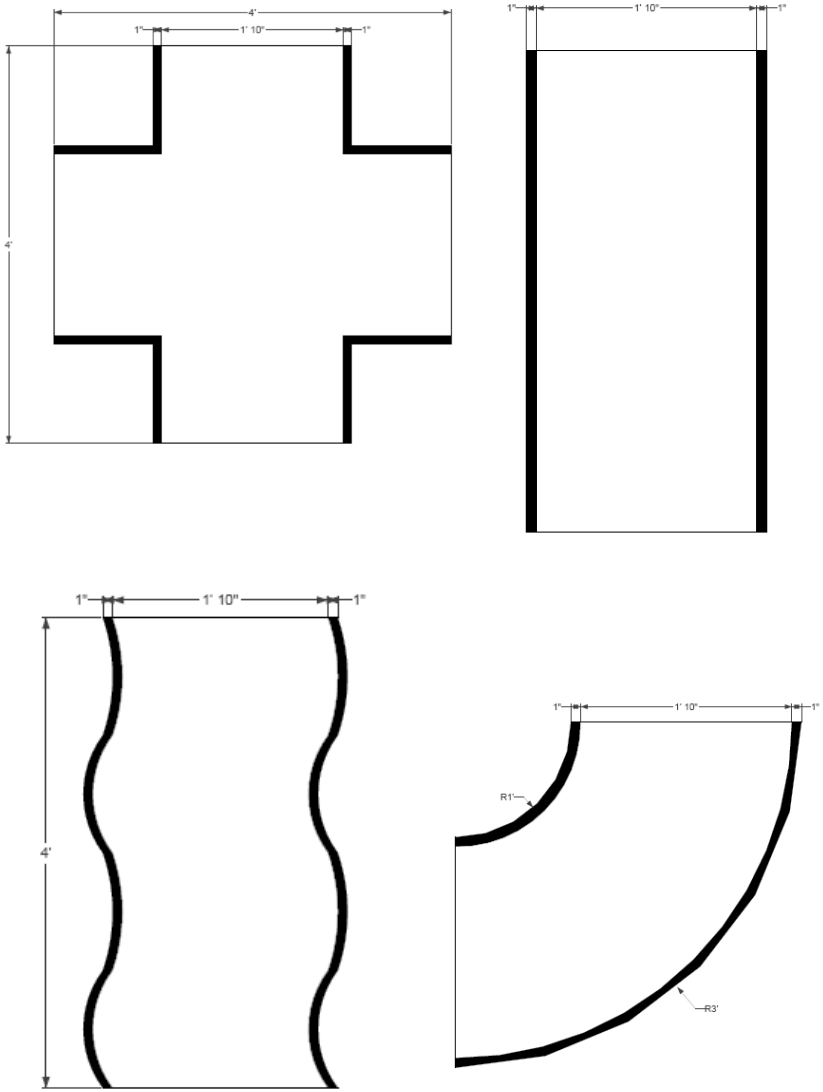
The Race Track

The race track consists of standardized parts. The exact layout of the track is not known and may change from race to race.

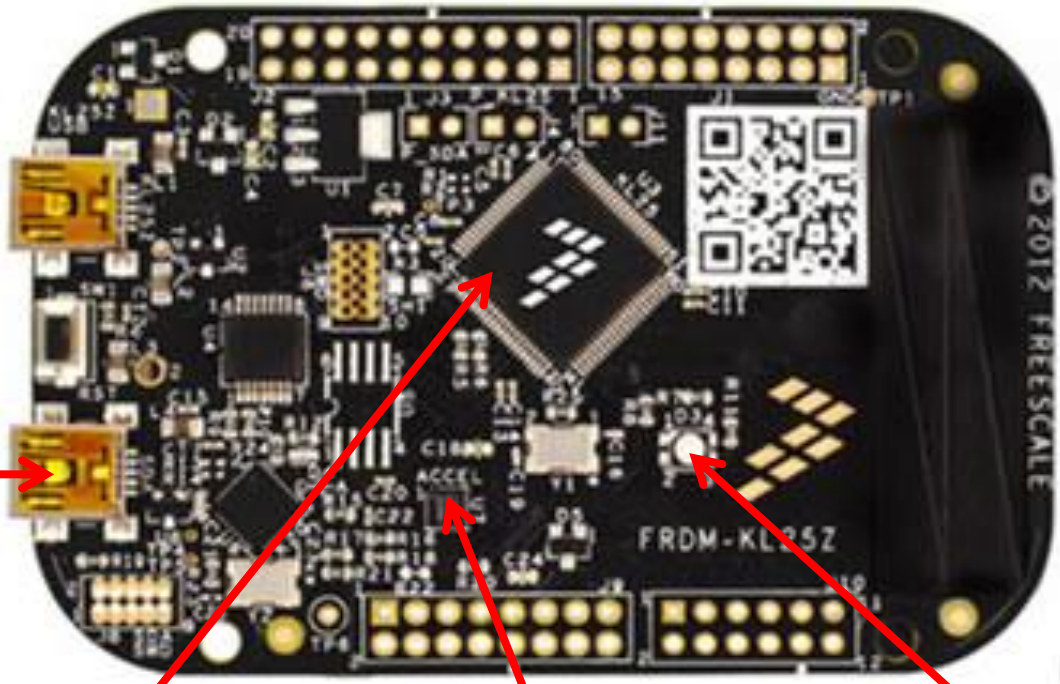
The road itself is white with a black line on the left and right side of the road. Additional elements may be: curves, crossovers, wiggly line, bumpers, hills, tunnel.

The track has an electronic time measure-ment system. The car has to go one round as fast as possible and has to stop after one round.

All details of the competition are described in the official rule book.



FRDM-KL25Z Microcontroller Module



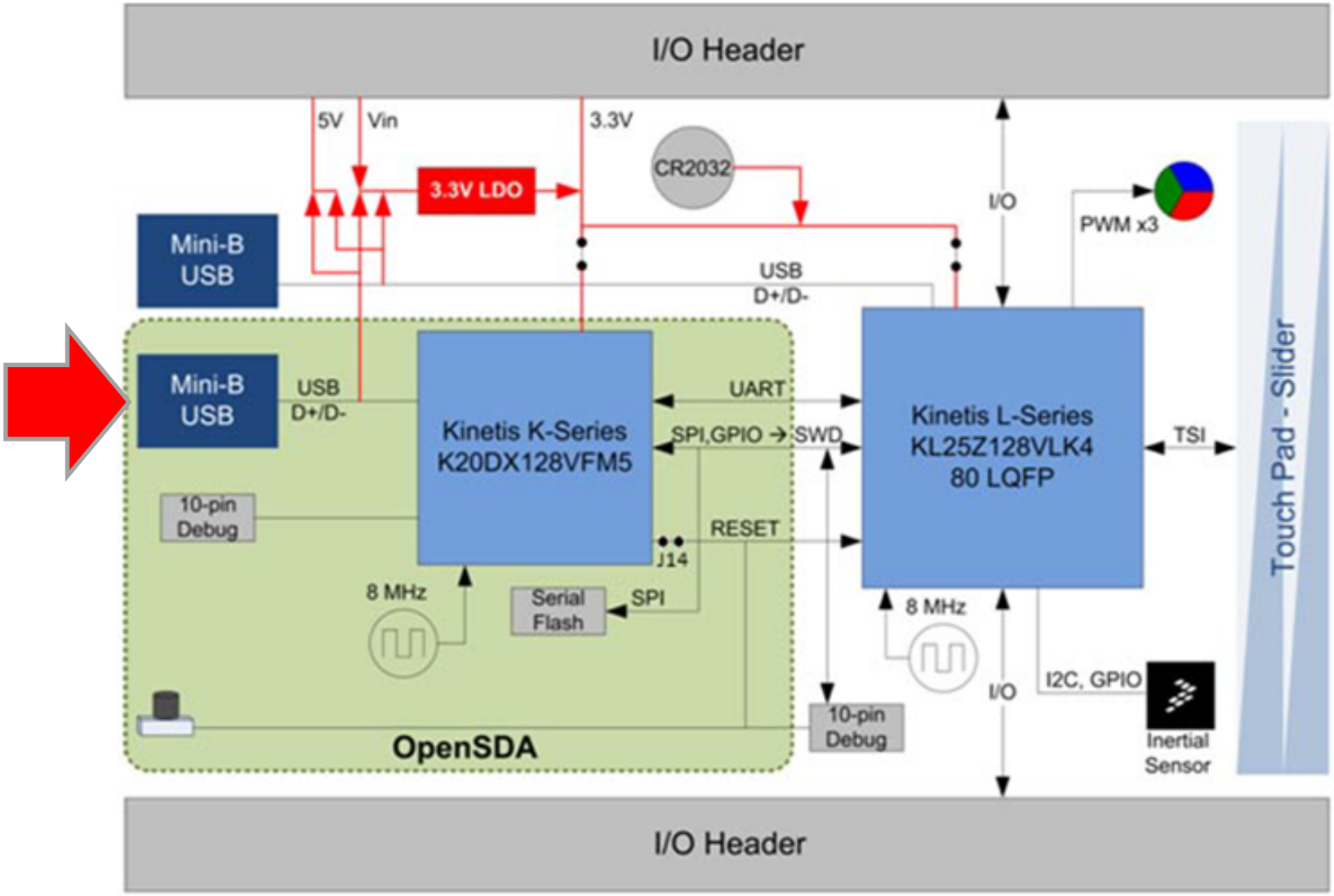
OpenSDA
Debug Port
and UART

KL25Z128VLK4
(48 MHz, 128 kB)

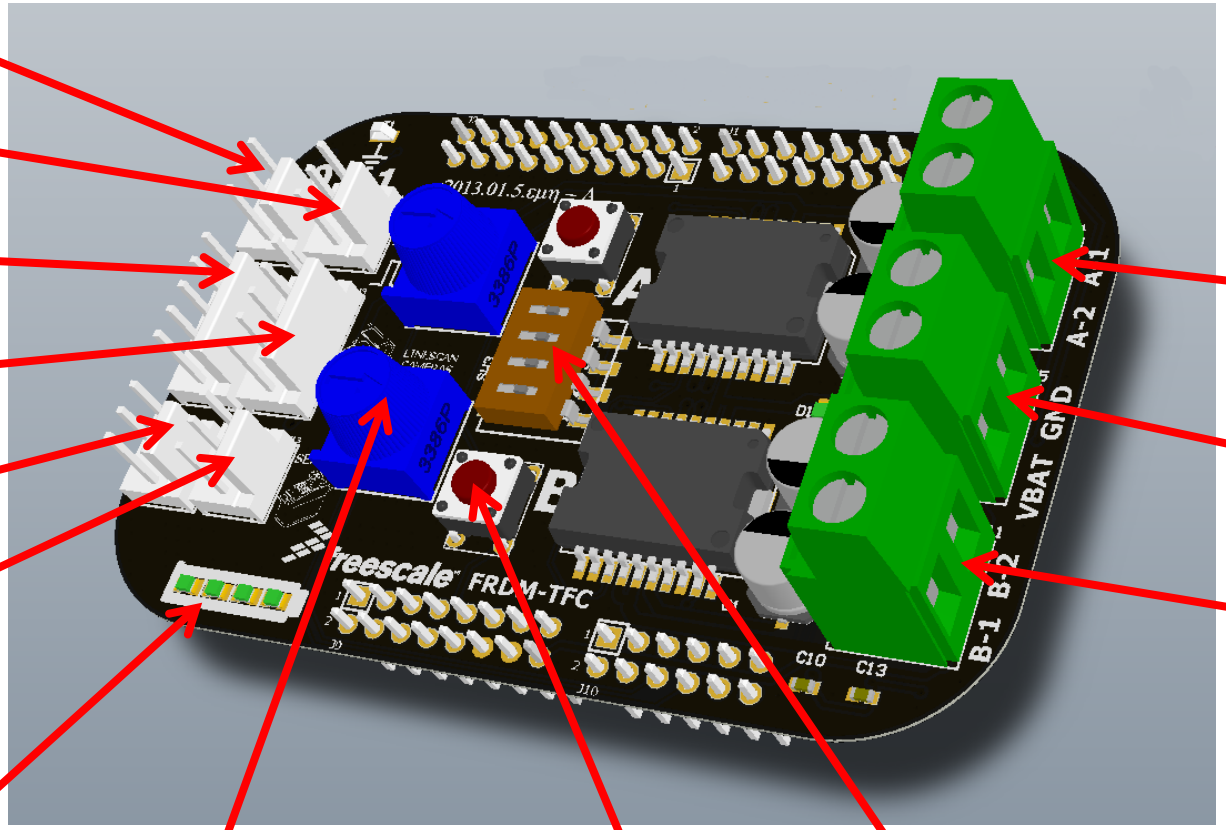
MMA8451Q
(3 axis sensor)

3 color LED

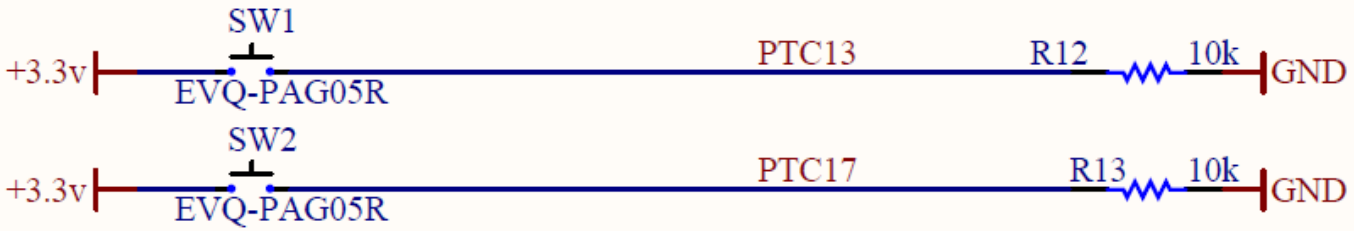
FRDM-KL25Z Microcontroller Module, Debug Interface



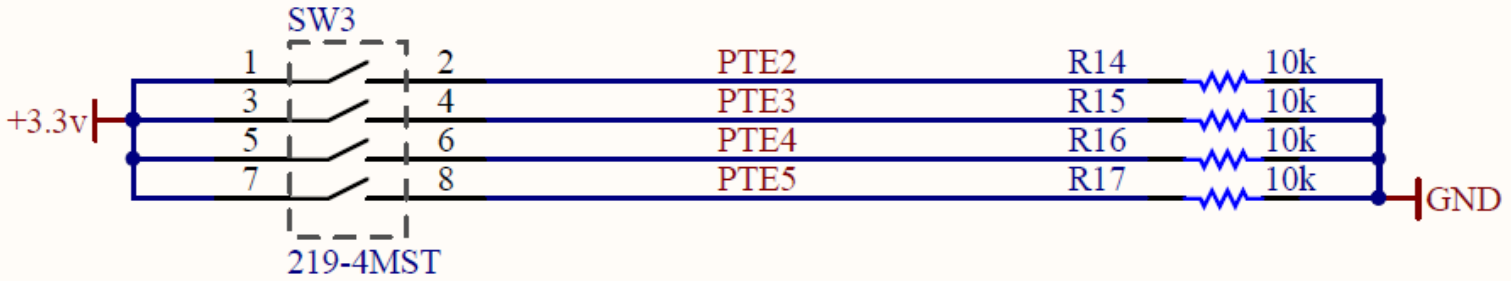
NXP Cup Car Standard Control Unit (TFC Motor Shield)



Switches

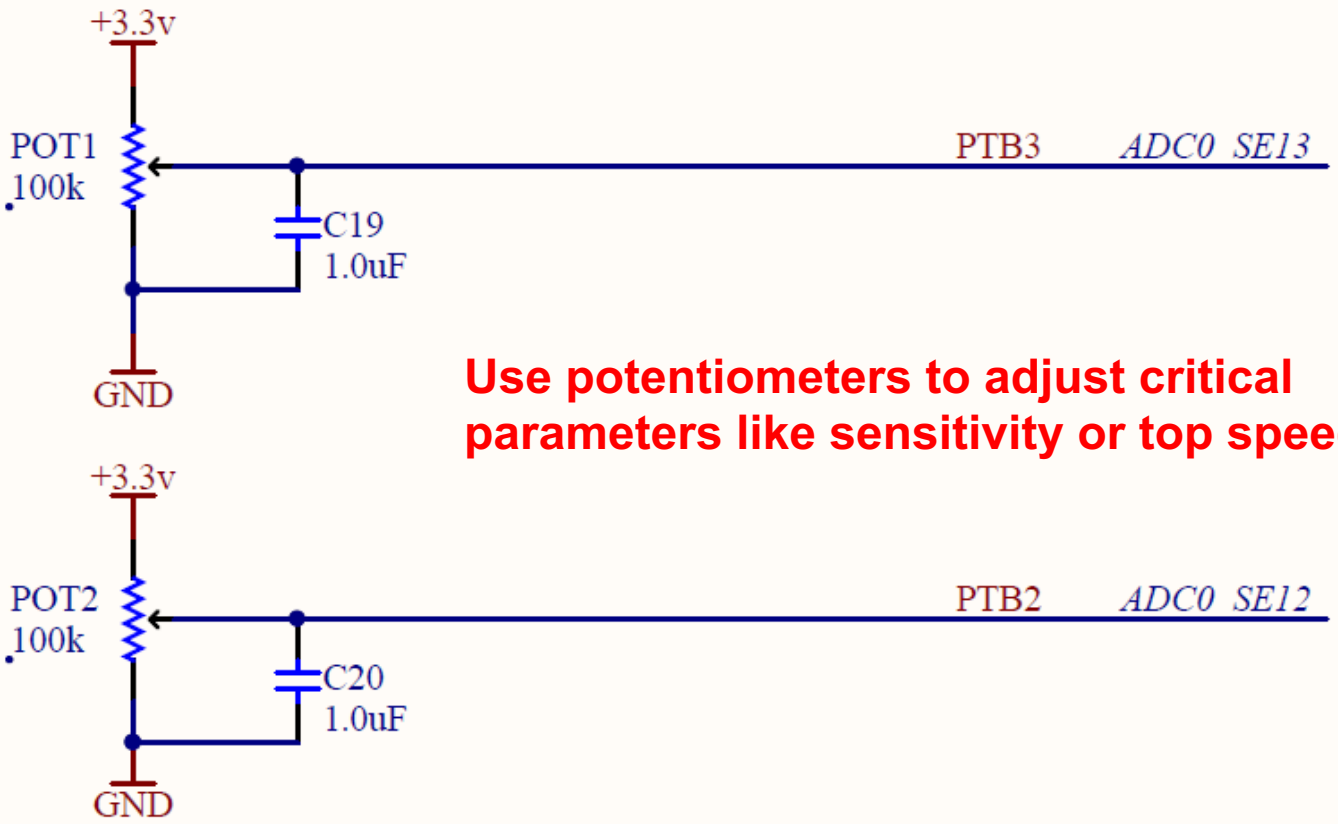


Use push buttons to start or stop the car.



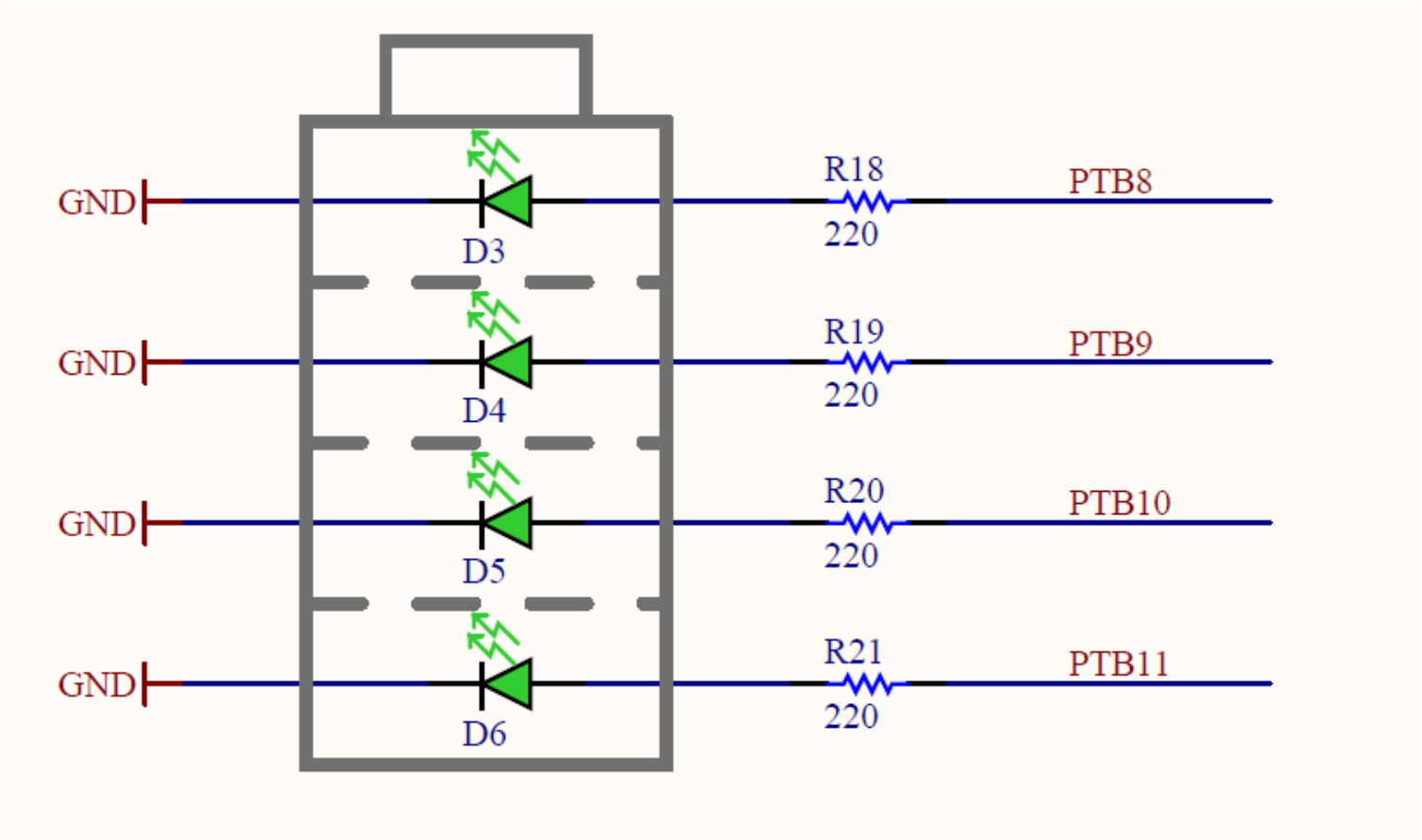
Use DIP switches to store different software options or strategies.

Potentiometers



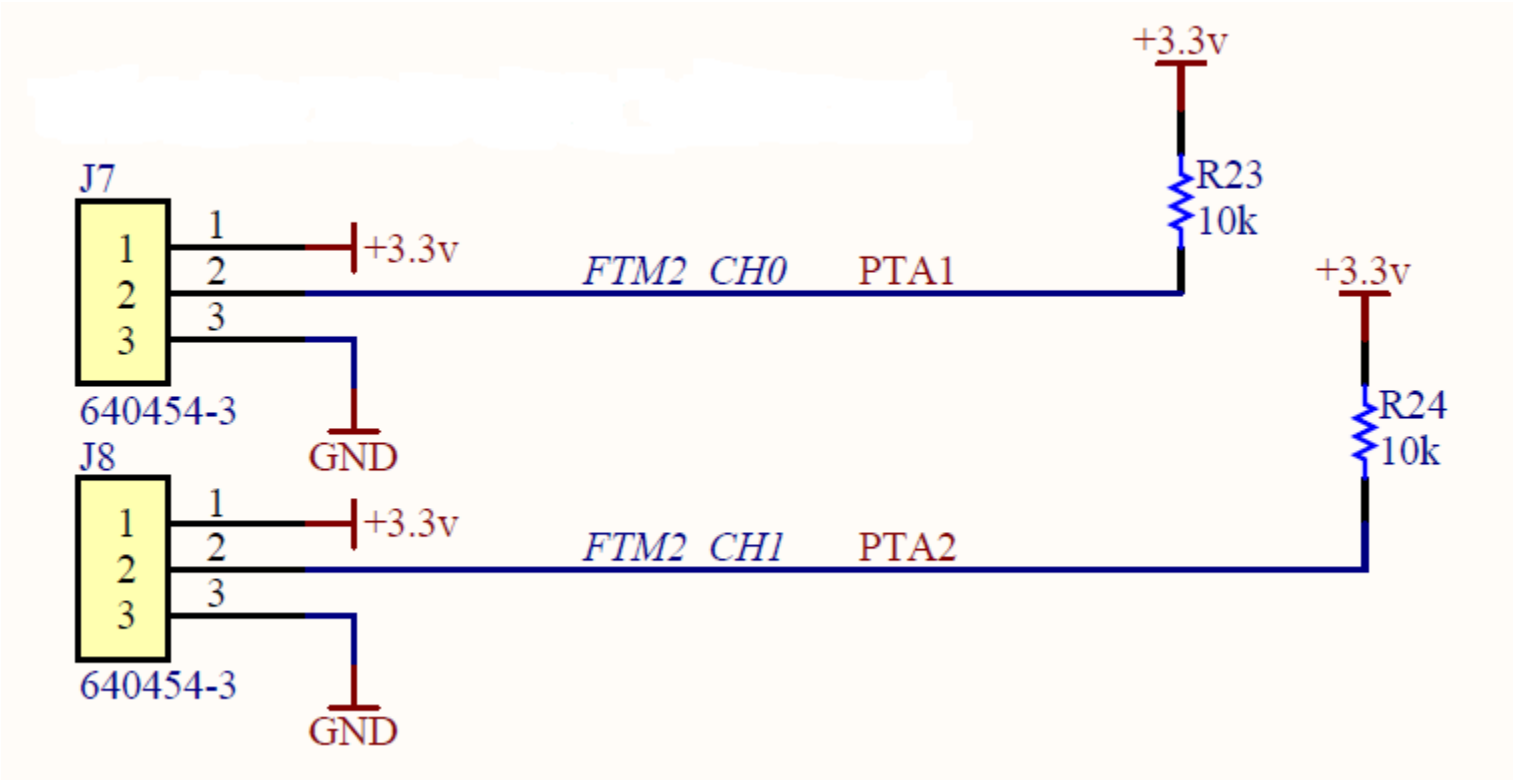
Use potentiometers to adjust critical parameters like sensitivity or top speed.

LEDs



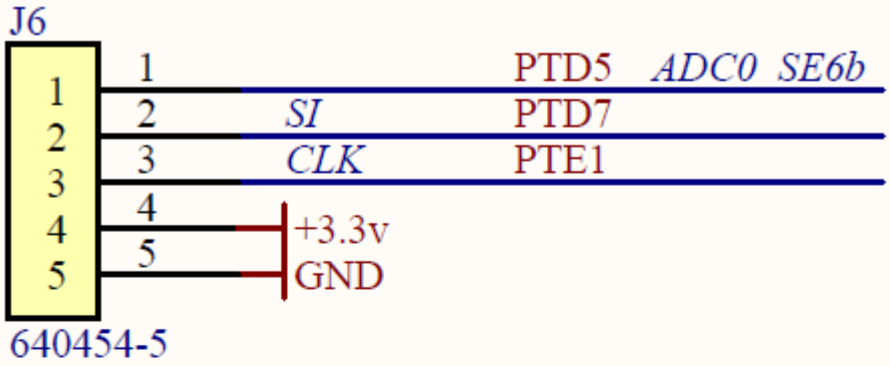
Use LEDs to display different status information of the sofware.

Speed Sensor Inputs

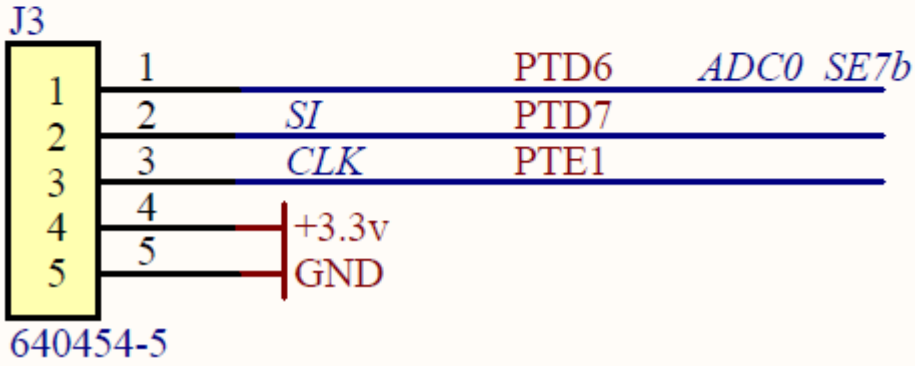


Use these sensor inputs for additional speed sensors (odometry)
- not used in the minimalistic software examples -

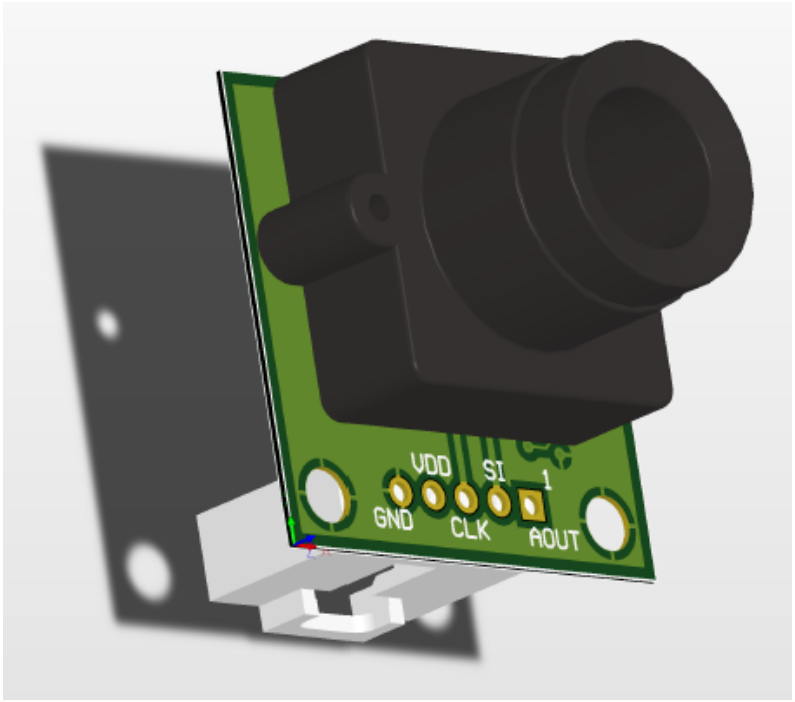
Line Scan Camera Interfaces

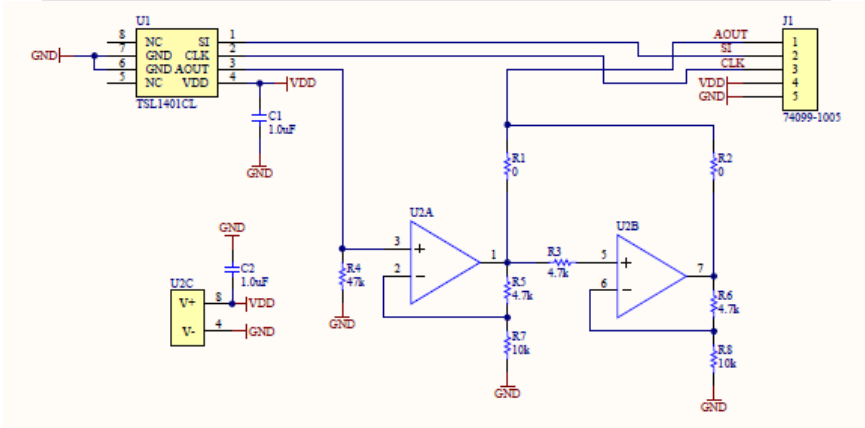


Just one line scan camera is used in the software examples.
The second camera interface can be used as well.



NXP Cup Line Scan Camera



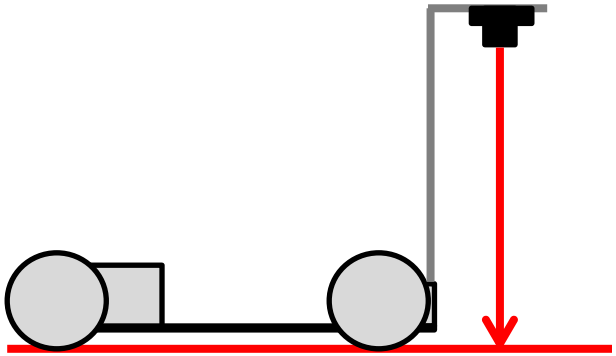


- **128-pixel linear image sensor**
- Focusable imaging lens
- 5-pin physical interface on PCB
- Simple three-pin MCU interface with analog pixel output
- Lens: 7.9mm focal length, manual focus, 12mm x 0.5mm thread
- Exposure time: 267µs to 68ms
- Resolution: 128 pixels
- Built-In amplifier stage to improve white/black differentiation.

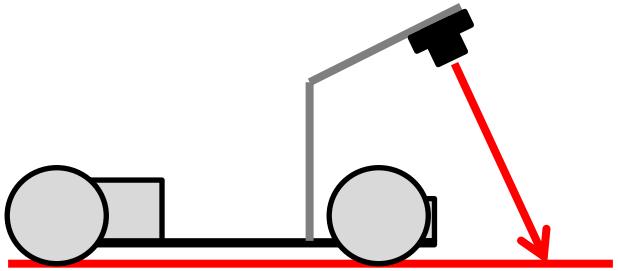
5 Pins:

GND	Ground
VDD	Supply Voltage 3.3 V
CLK	Clock
SI	Scan Impulse
AOUT	Analog Out

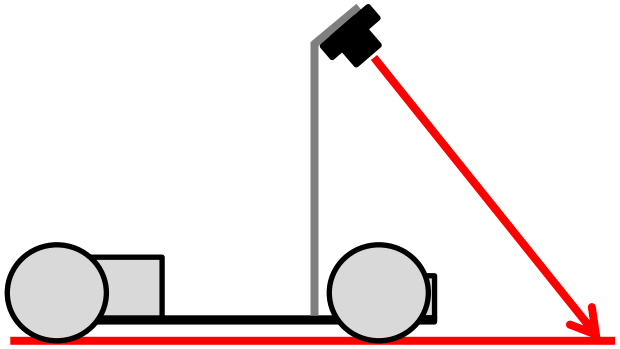
Mounting the Camera: Position is Key!



Little foresight

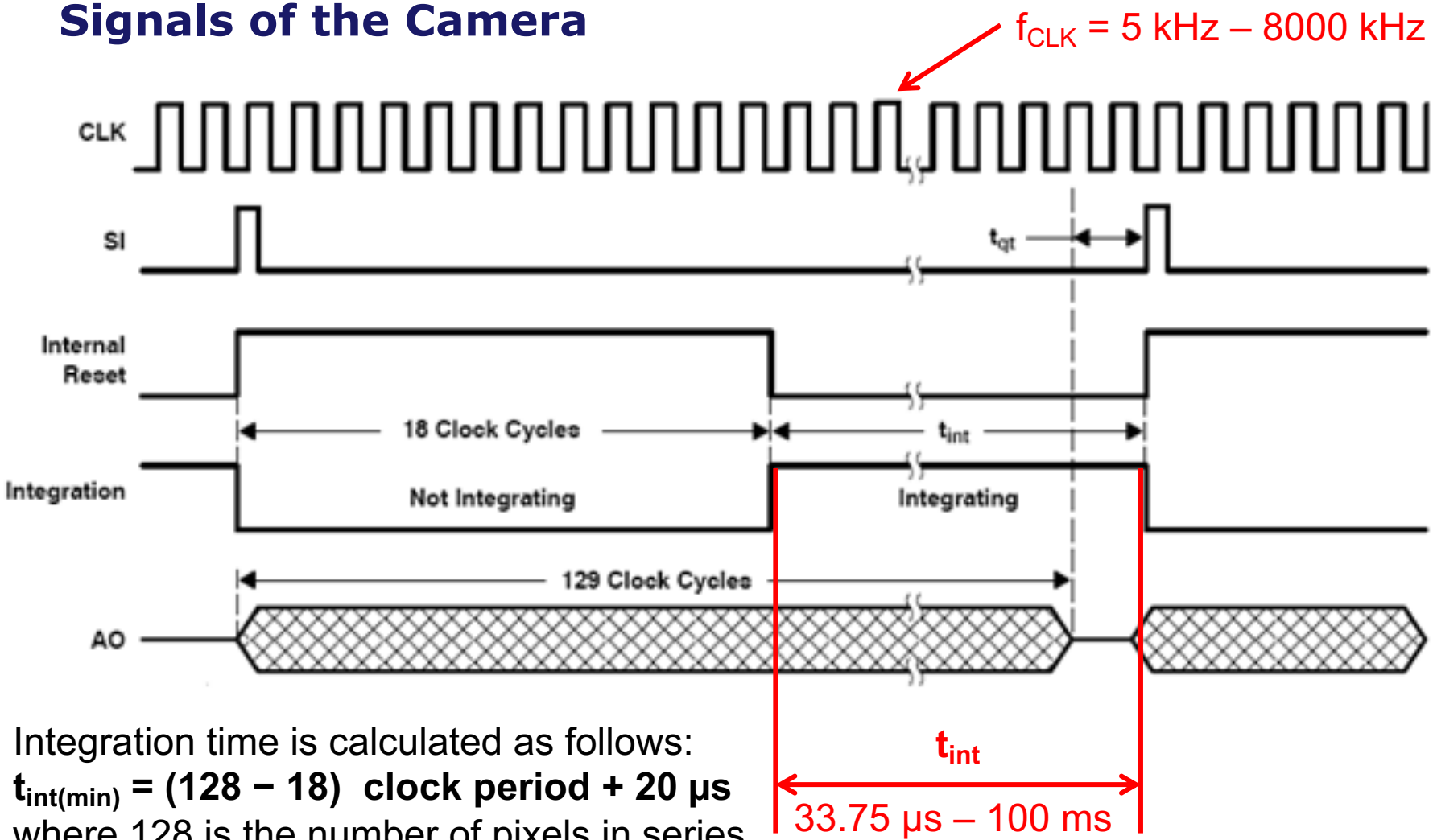


Medium foresight



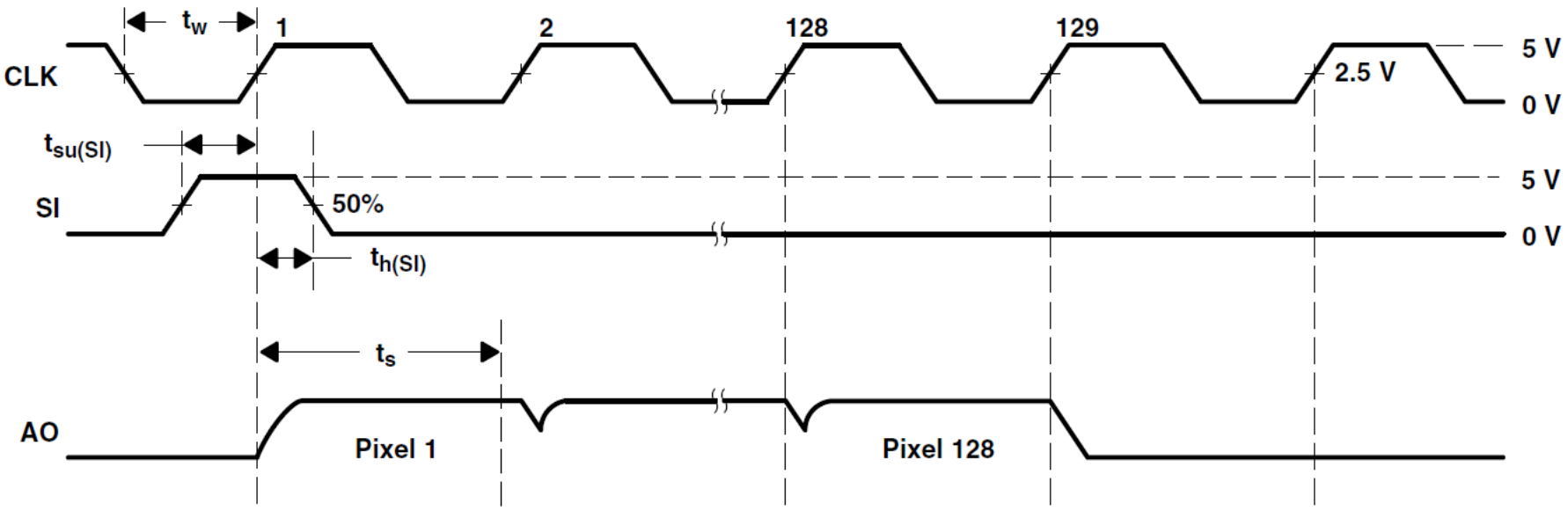
Large foresight

Signals of the Camera



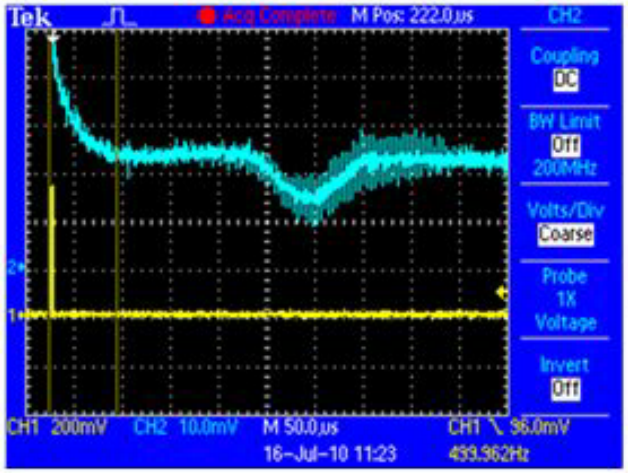
Integration time is calculated as follows:
$t_{int(min)}$ = (128 − 18) clock period + 20 µs
where 128 is the number of pixels in series,
18 is the required logic setup clocks, and
20 µs is the pixel charge transfer time (t_{qt}).

Details of the Camera Signals

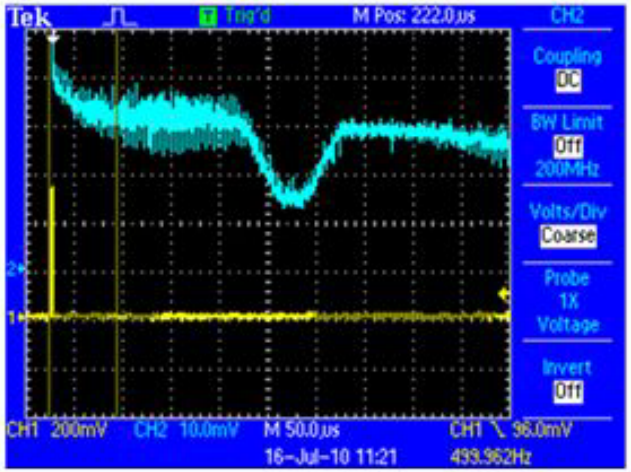


The clock signal (**CLK**) and the sync impulse (**SI**) should be delivered from the microcontroller. Since the MCU does not have a dedicated camera interface, GPIO pins should be used. The analog output signal (**AO**) of the camera should be read by the ADC of the microcontroller.

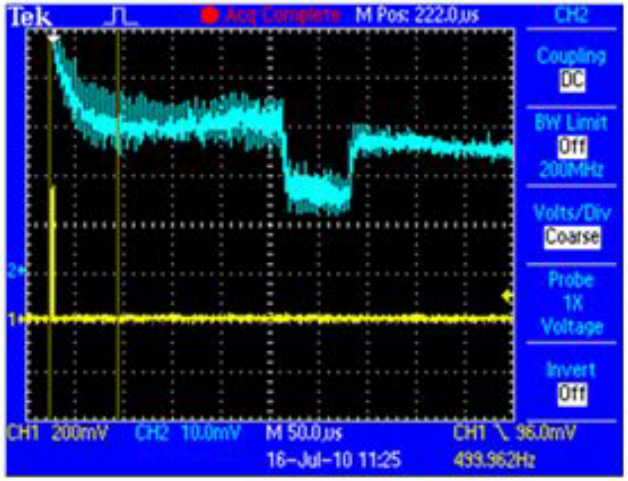
Line Scan Camera signals shown on an Oscilloscope



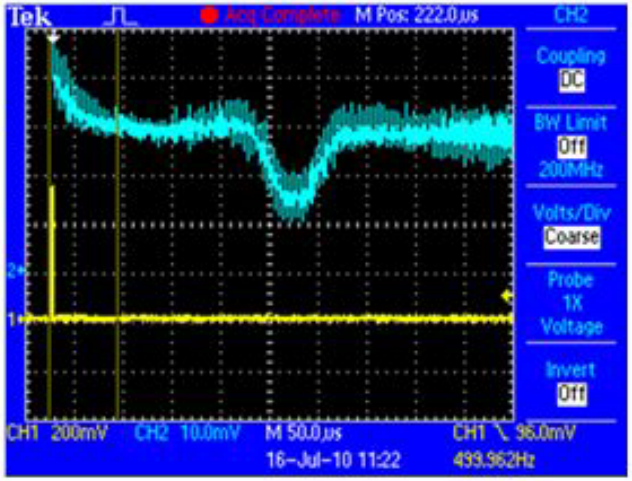
In this picture, the lens was almost out of the mount, therefore, it was not focused



This picture was taken after screwing a little more the lens on the mount

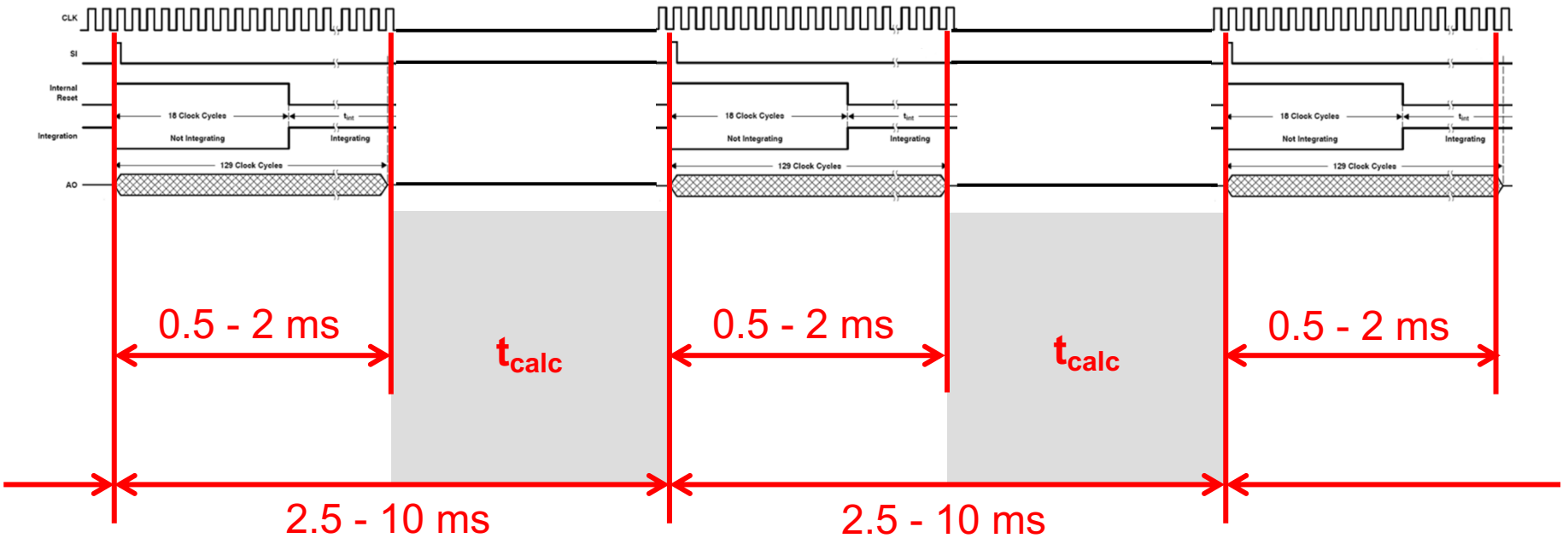


In this picture the lens was completely focused that's why the figure of the AO seems like a square



In this picture the lens was screwed a little more after being focused, therefore it became unfocused again

Timing of the Camera Signals



50 ms Integration time -> Voltage level: 3.3 V
10 ms Integration time -> Voltage level: 1.2 V
5 ms Integration time -> Voltage level: 0.8 V
2.5 ms Integration time -> Voltage level: 0.5 V

t_{calc}: time for calculation

Maximum Clock of ADC

Symbol	Description	Conditions	Min.	Typ.[1]	Max.	Unit
V_{DDA}	Supply voltage	Absolute	1.71	—	3.6	V
f_{ADCK}	ADC conversion clock frequency	≤ 13 bit modes	1.0	—	18.0	MHz
f_{ADCK}	ADC conversion clock frequency	16 bit modes	2.0	—	12.0	MHz

The ADC should work as fast as possible. Use 8 bit or 10 bit resolution.

Recommendations for using the Line Scan Camera

1. Check the camera readings (using an example software or an oscilloscope).
2. Mount the camera in a way that you have a good visibility and a clear view of the track.
3. You may vary the exposure time, so you can adapt the camera reading to the lighting conditions.
4. Cover the camera (not the lense) with tape, to avoid the influence of surrounding light.
5. The frame rate of the line scan camera should be at least 100 frames per second, or even higher.
6. You can use two line scan cameras on the car.
7. If necessary, use additional illumination of the track.

Freescale Community / NXP Community

https://community.nxp.com/groups/tfc-emea

Pro Tips:

#1 - Avoid Light Noise

#2 - Know your Settings

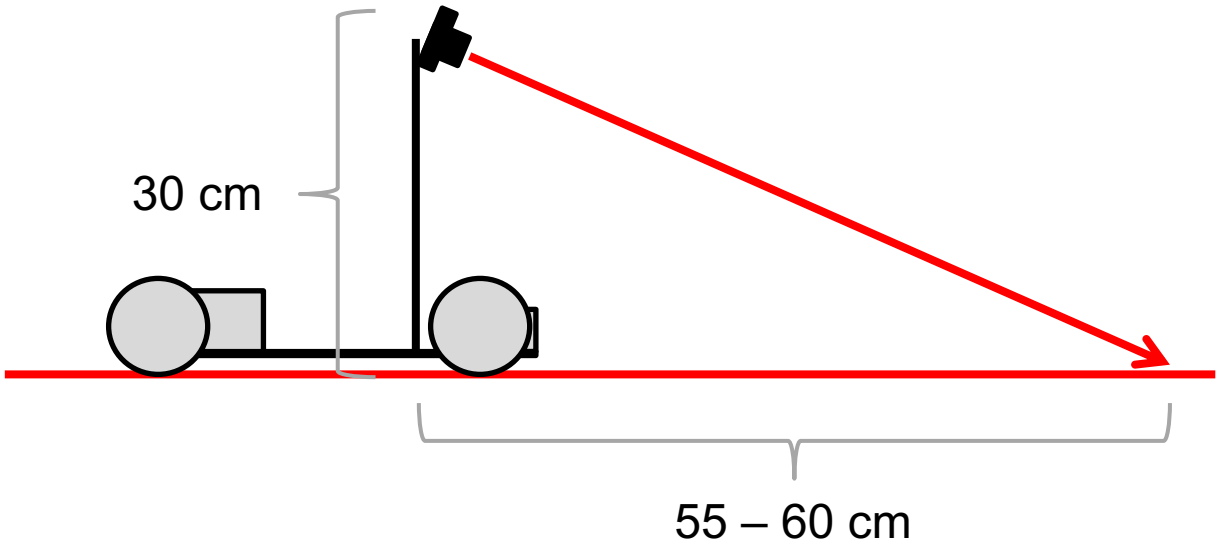
#3 - Know your Zone

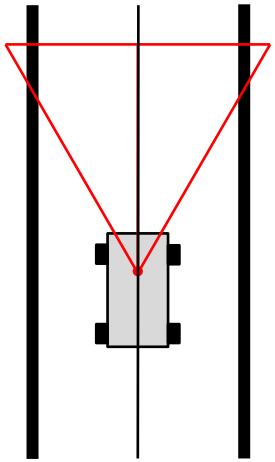
#4 - Buffer Data
Since the camera can read the line very quickly while the servo can only update every 10-20 ms, there are multiple camera reads before the servo can update, if you are reading the camera fast and then overriding without saving them in some form then those camera reads are being wasted and are better off not having occurred. What can help is to create some sort of filter by bringing new values into an array with previous values and preforming some sort of averaging.

Recommendation: Adjustment of the Line Scan Camera

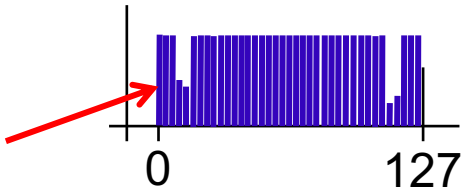
Adjust the line scan camera in a way, that you see the both black lines on the left and right side of the race track. Use the oscilloscope or the example software.

Use a track segment with an adjustment pattern.

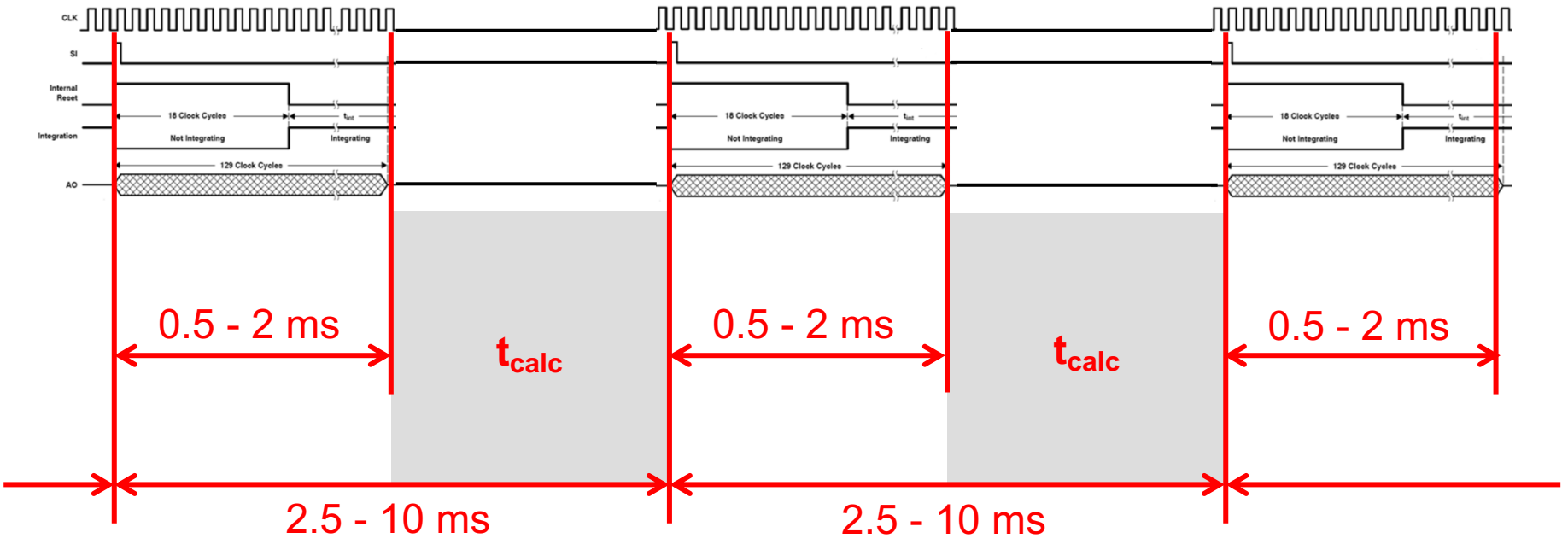




line scan camera data:
array of 128 values, every value represents a pixel.



Recommendation: Frame Rate of the Line Scan Camera



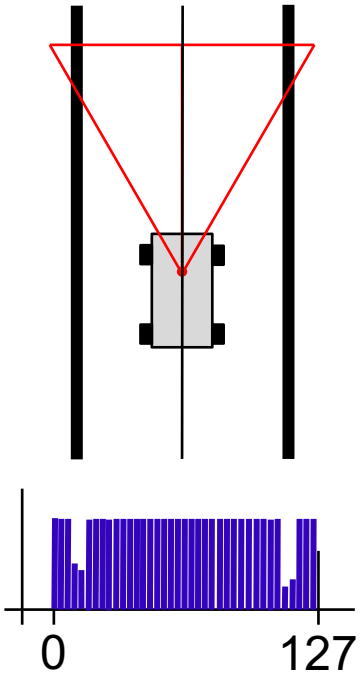
recommended →

20 Hz -> 50 ms Integration time -> Voltage level: 3.3 V
100 Hz -> 10 ms Integration time -> Voltage level: 1.2 V
200 Hz -> 5 ms Integration time -> Voltage level: 0.8 V
400 Hz -> 2.5 ms Integration time -> Voltage level: 0.5 V

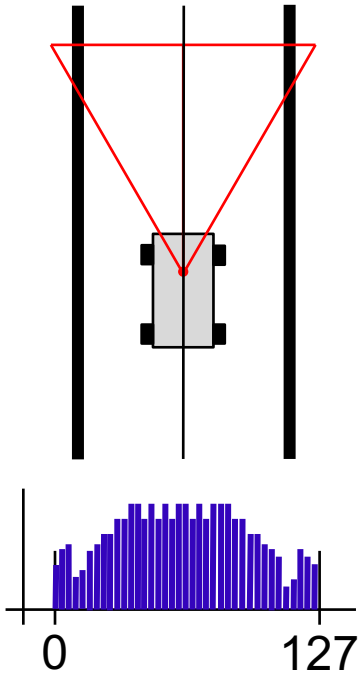
Recommendation: Robust Line Identification

Design the identification of the black lines very robust, so that you can deal with all kinds of various illuminations.

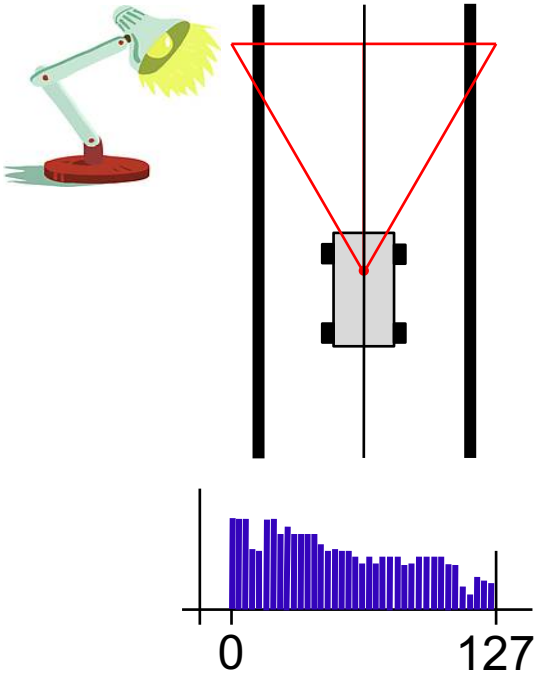
ideal conditions



average conditions



illumination from the left



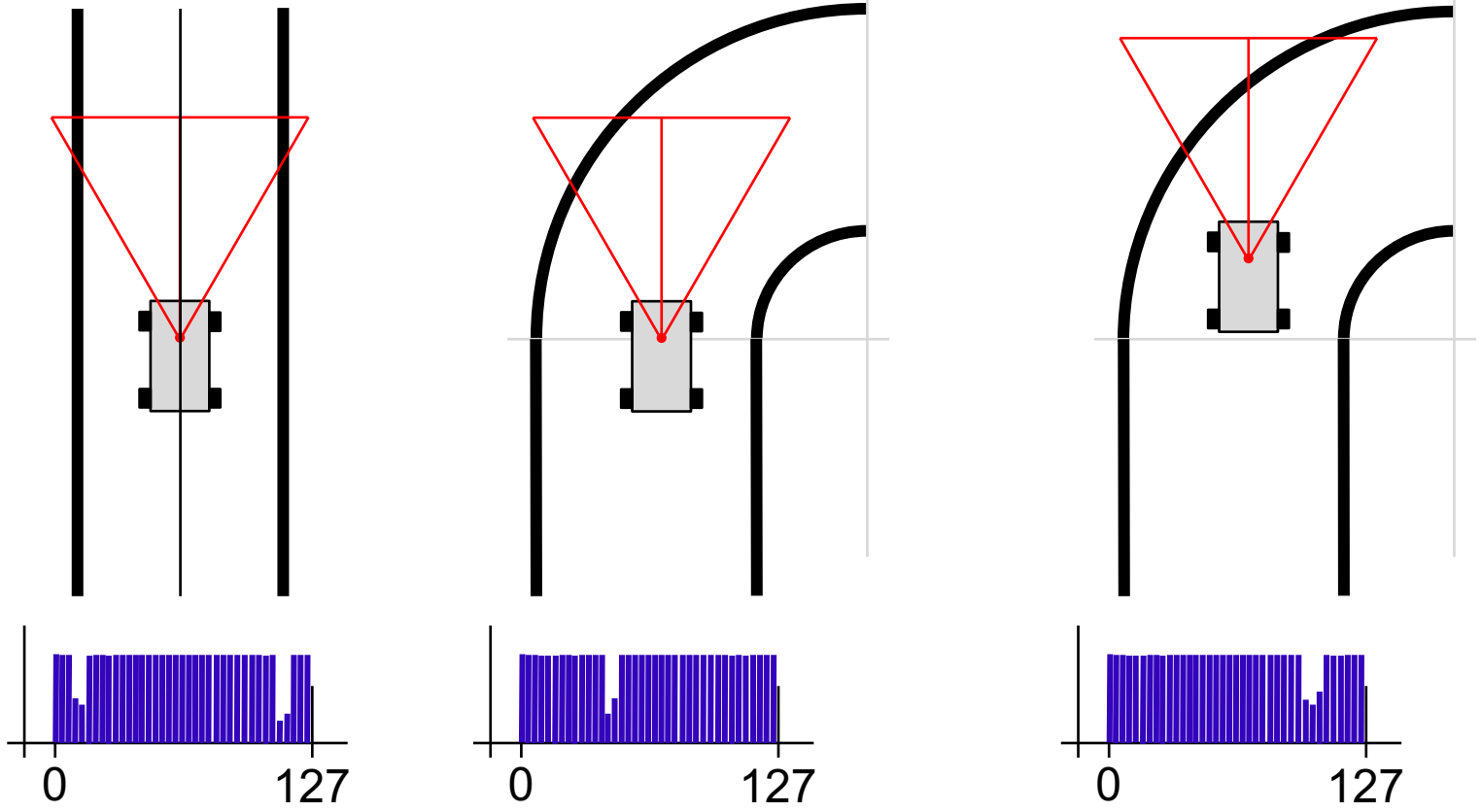
Recommendation: use differential line recognition

In order to create a robust line recognition, you should use a differential signal. That means, that not the absolute pixel value is used, but the difference of values between two pixels.

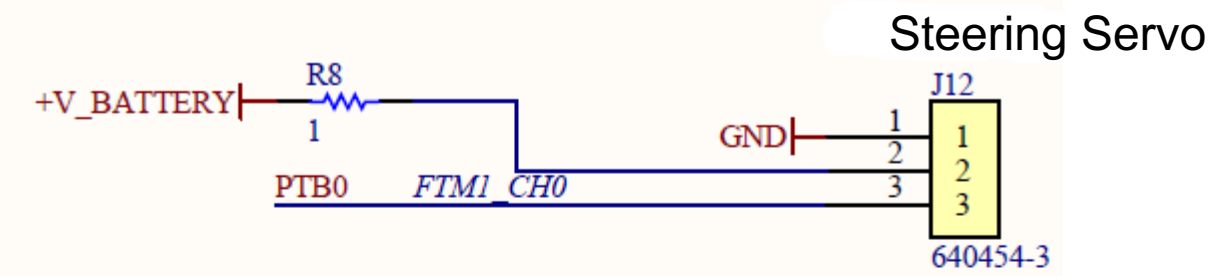
```
// Find black line on the right side
for(i=64;i<127;i++)// calculate difference between the pixels
{
ImageDataDifference[i] = ImageData[i] - ImageData[i+1];
}
CompareData = 3;
for(i=64;i<127;i++)
{
   if (ImageDataDifference[i] > CompareData)
   {
   CompareData = ImageDataDifference[i];
   BlackLineRight = i;
   }
}
```

Recommendation: Identification of left / right turn

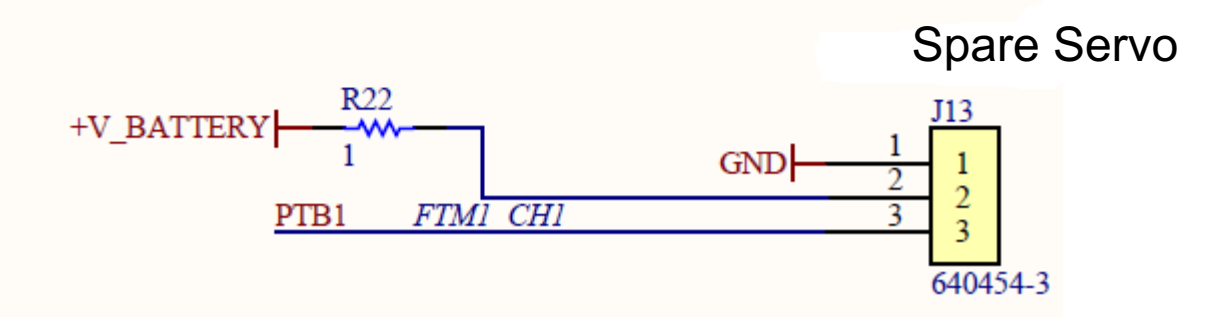
Use plausibitity check / historic data to identify if you are in a left turn or right turn.



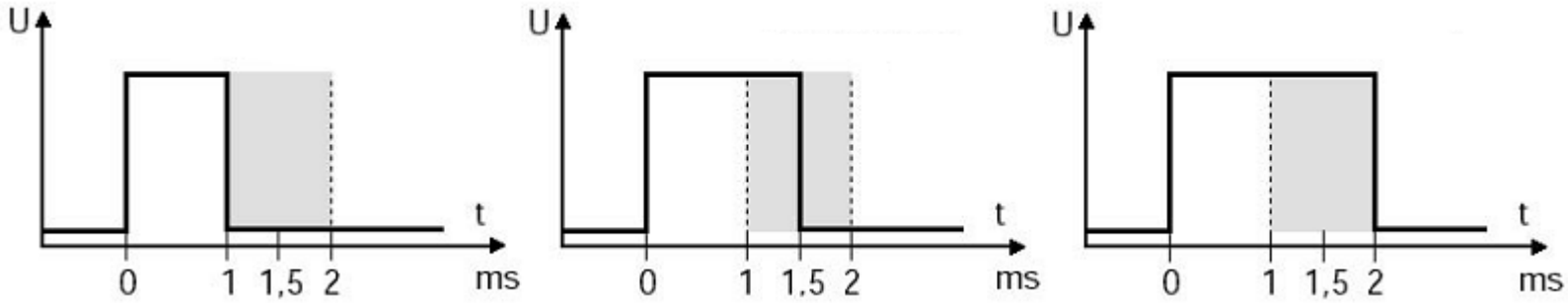
Servo Outputs



Just one steering servo motor is used in the software examples.
The second servo motor interface can be used as well.



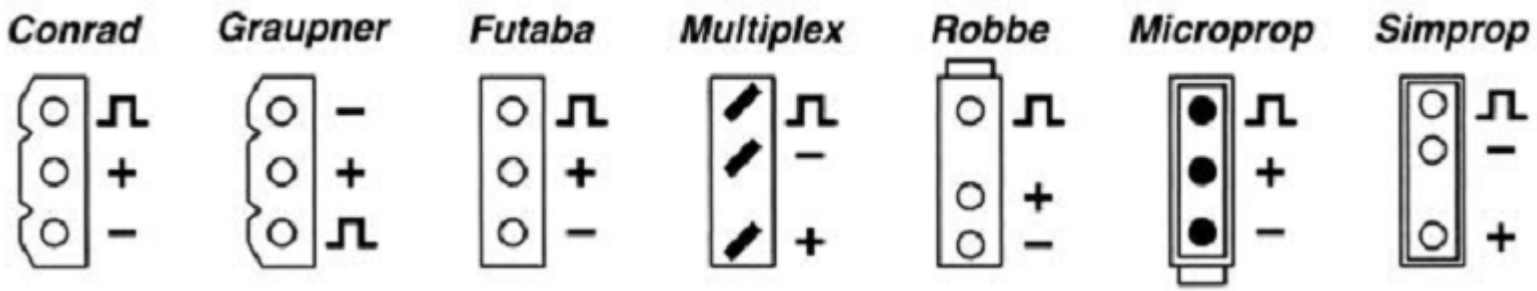
Control of Servo Motors



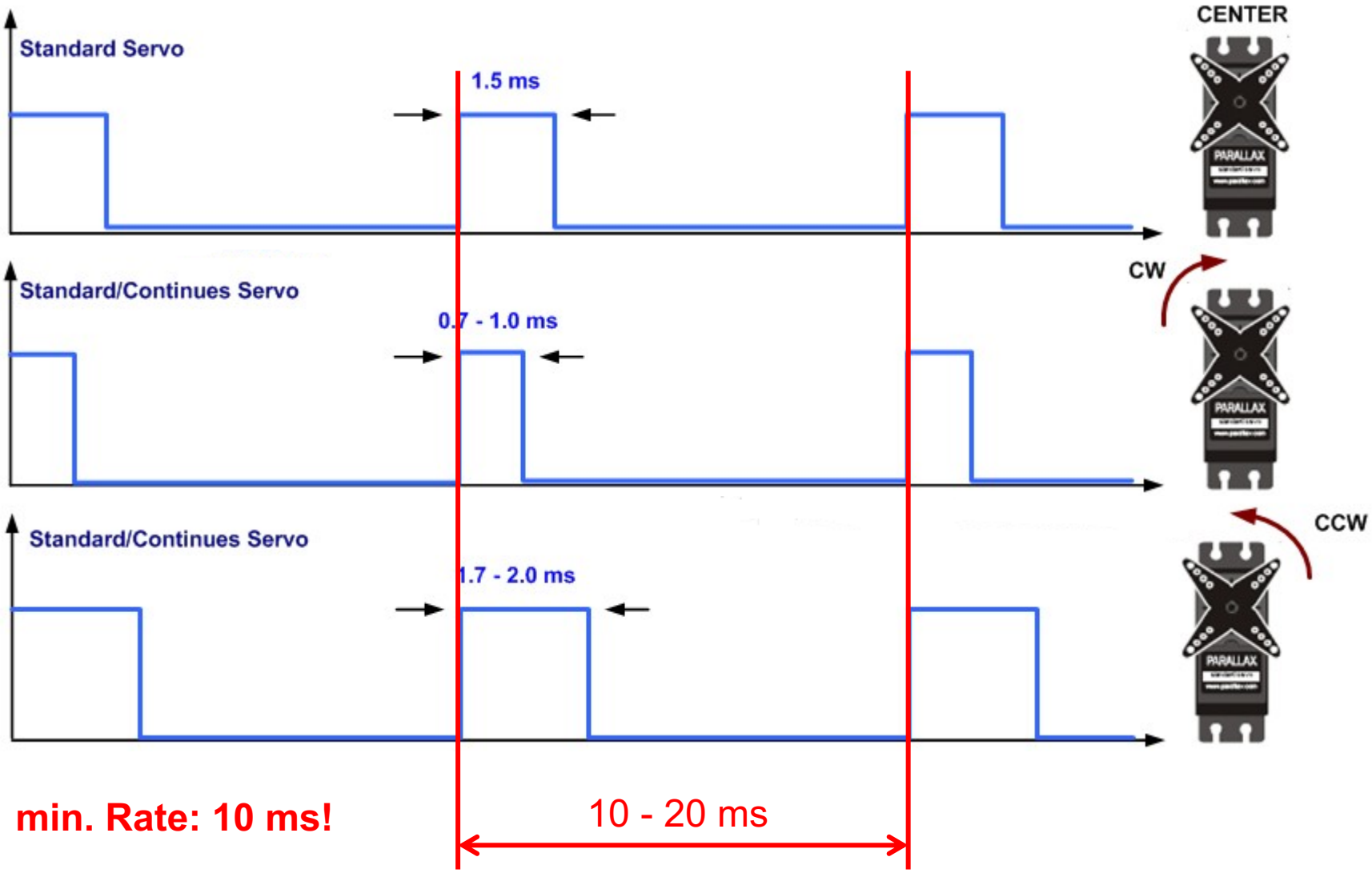
Minimum Neutral Maximum

The servo motor should receive a new impulse all 10 - 20 ms. The duration of the impulse should be between 1 ms and 2 ms and defines the position of the servo.

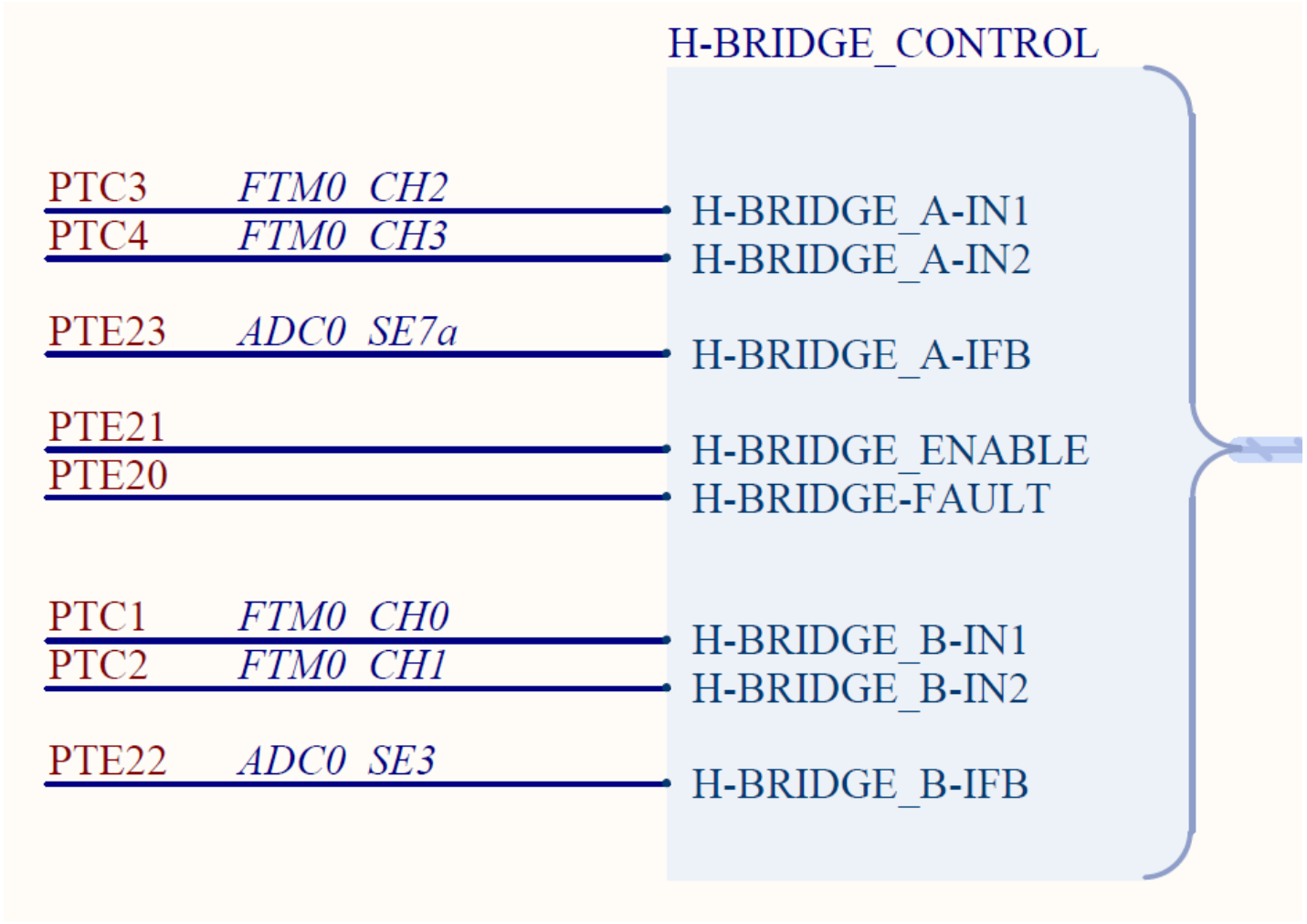
We should deliver a new servo motor impulse all 10 - 20 ms (50 - 100 Hz).



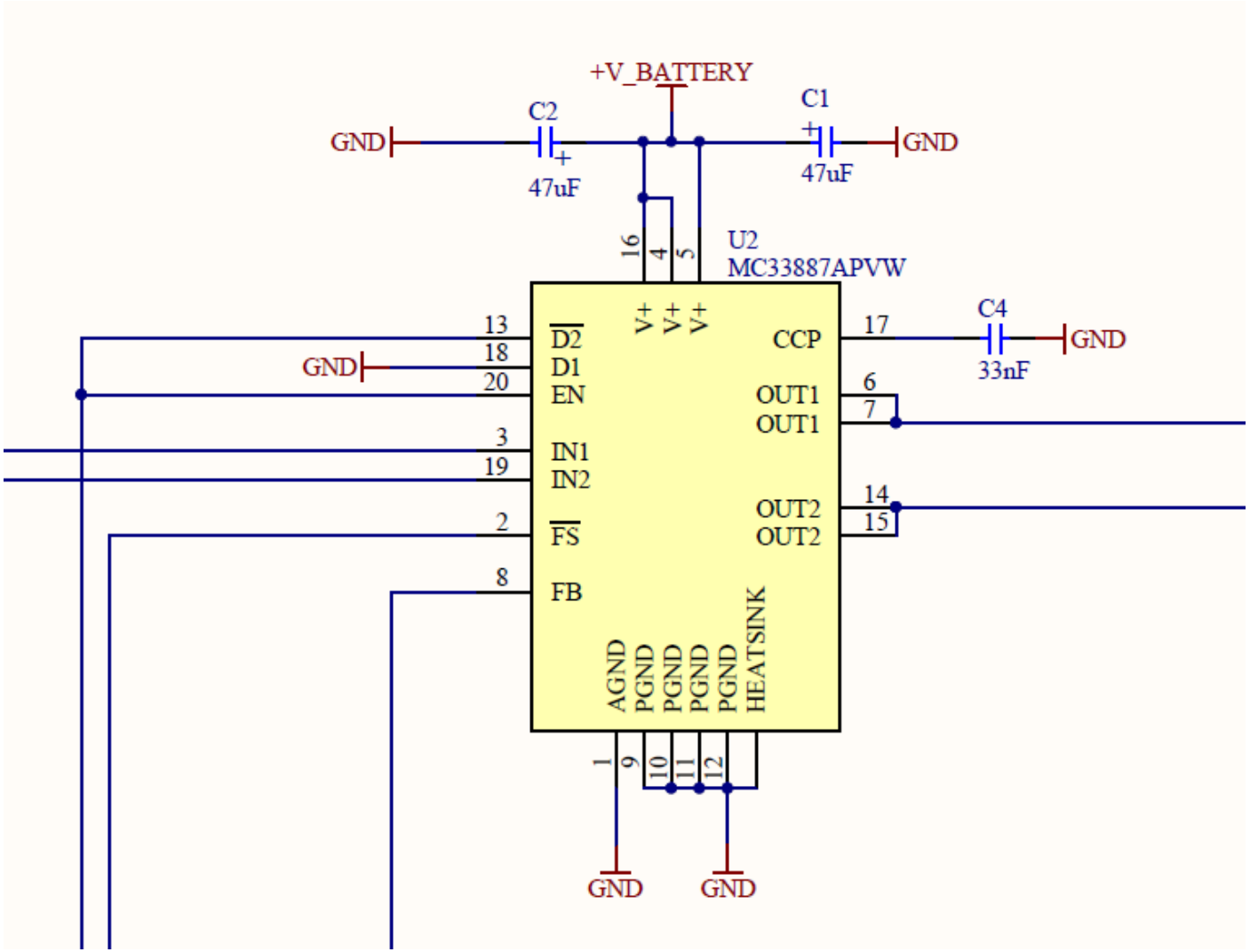
Update Rate of Servo Motor



Main Drive Motors



MC33887 Motor Driver



MC33887 H-Bridge

Freescale Semiconductor
Technical Data

Document Number: MC33887
Rev. 16.0, 10/2012

RoHS

5.0 A H-Bridge with Load Current Feedback

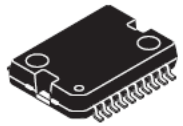
33887

H-BRIDGE

The 33887 is a monolithic H-Bridge Power IC with a load current feedback feature making it ideal for closed-loop DC motor control. The IC incorporates internal control logic, charge pump, gate drive, and low $R_{DS(ON)}$ MOSFET output circuitry. The 33887 is able to control inductive loads with continuous DC load currents up to 5.0 A, and with peak current active limiting between 5.2 A and 7.8 A. Output loads can be pulse width modulated (PWM-ed) at frequencies up to 10 kHz. The load current feedback feature provides a proportional (1/375th of the load current) constant-current output suitable for monitoring by a microcontroller's A/D input. This feature facilitates the design of closed-loop torque/speed control as well as open load detection.

A Fault Status output pin reports undervoltage, short circuit, and overtemperature conditions. Two independent inputs provide polarity control of two half-bridge totem-pole outputs. Two disable inputs force the H-Bridge outputs to tri-state (exhibit high-impedance).

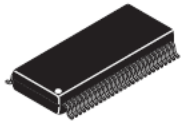
The 33887 is parametrically specified over a temperature range of -40°C ≤ T_A ≤ 125°C and a voltage range of 5.0 V ≤ V+ ≤ 28 V. Operation with voltages up to 40 V with derating of the specifications.



VW SUFFIX (Pb-FREE)
98ASH70702A
20-PIN HSOP

FK SUFFIX
98ASA10583D
36-PIN PQFN

Bottom View



EK SUFFIX (Pb-FREE)
98ASA10506D
54-PIN SOICW-EP

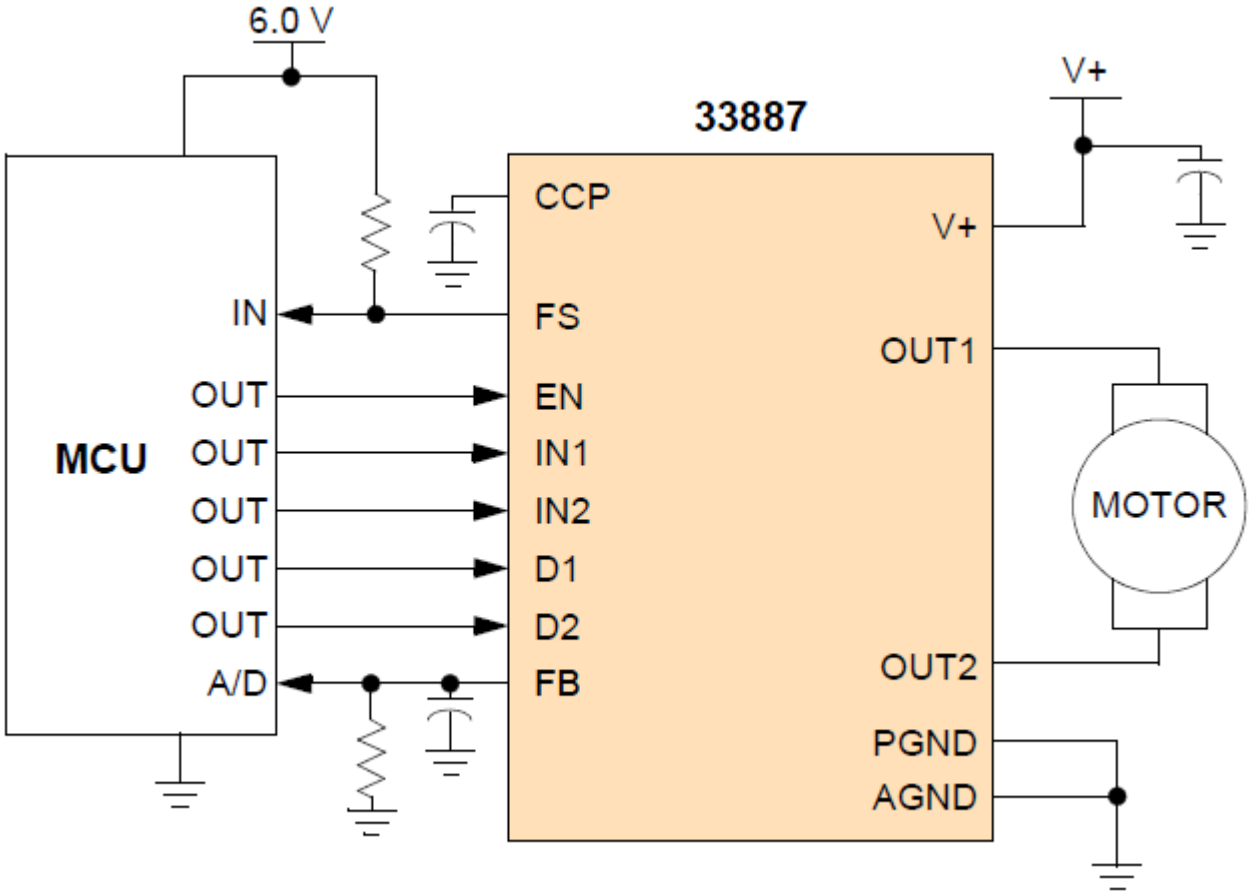
Features

- Fully specified operation 5.0 V to 28 V
- Limited operation with reduced performance up to 40 V
- 120 mΩ $R_{DS(ON)}$ Typical H-Bridge MOSFETs
- TTL/CMOS Compatible Inputs
- PWM Frequencies up to 10 kHz
- Active Current Limiting (Regulation)
- Fault Status Reporting
- Sleep Mode with Current Draw ≤50 µA (Inputs Floating or Set to Match Default Logic States)

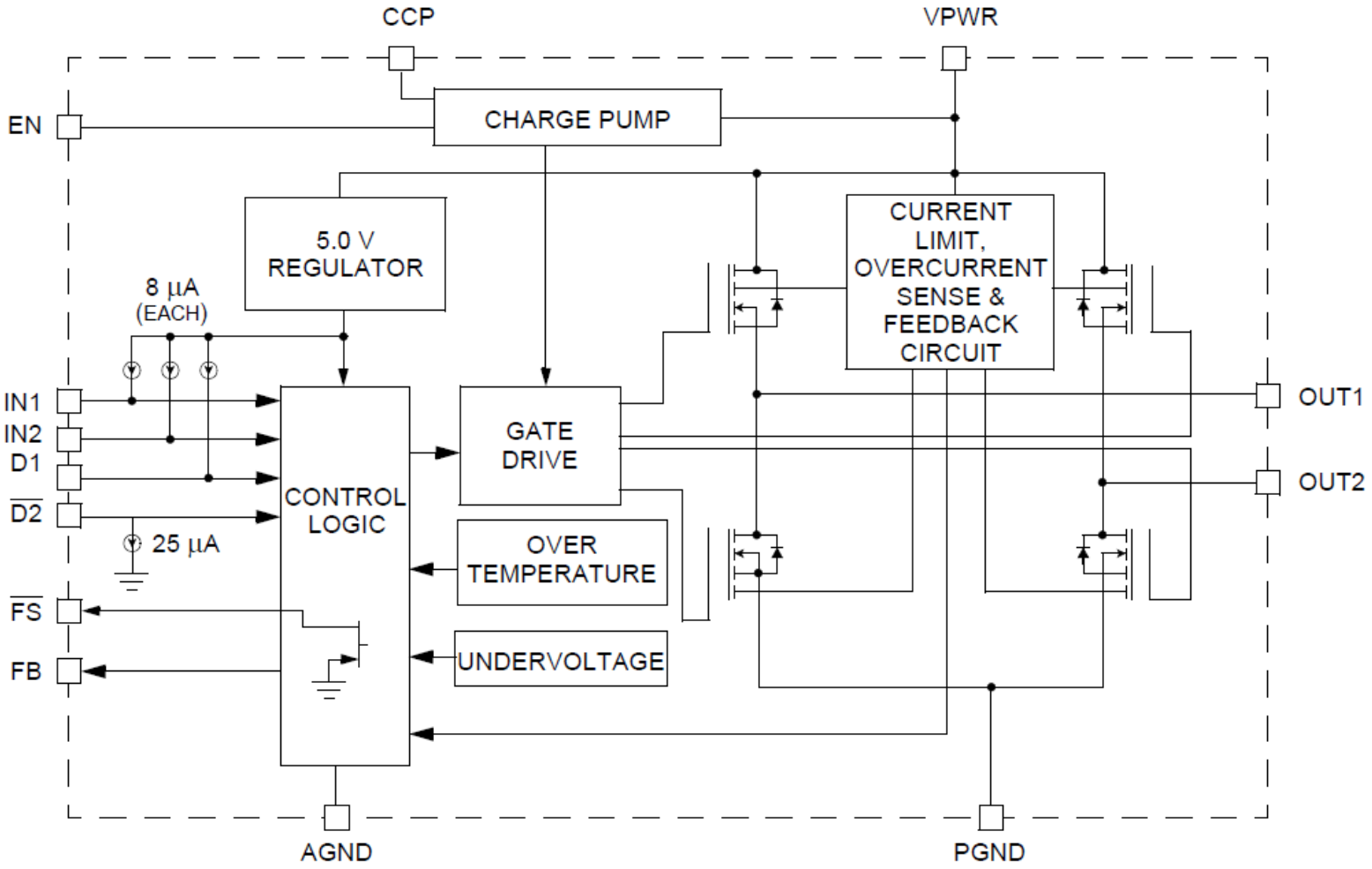
ORDERING INFORMATION

Device	Temperature Range (T_A)	Package
MC33887APVW/R2	-40°C to 125°C	20 HSOP
MC33887PFK/R2		36 PQFN
MC33887PEK/R2		54 SOICW-EP

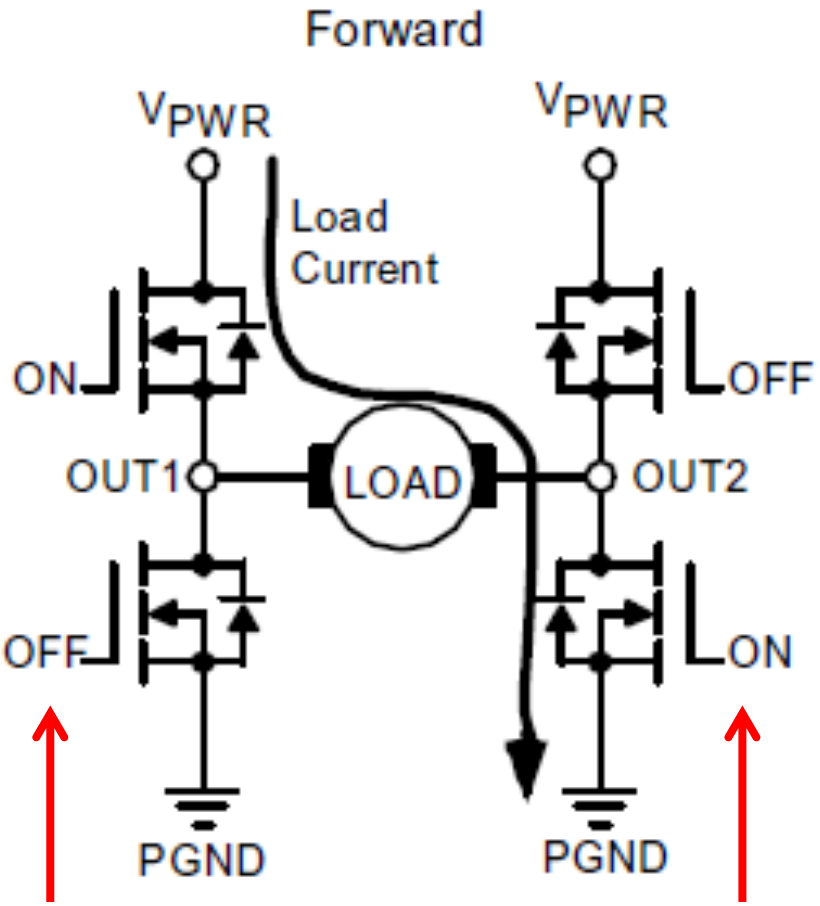
MC33887 Simplified Application Diagram

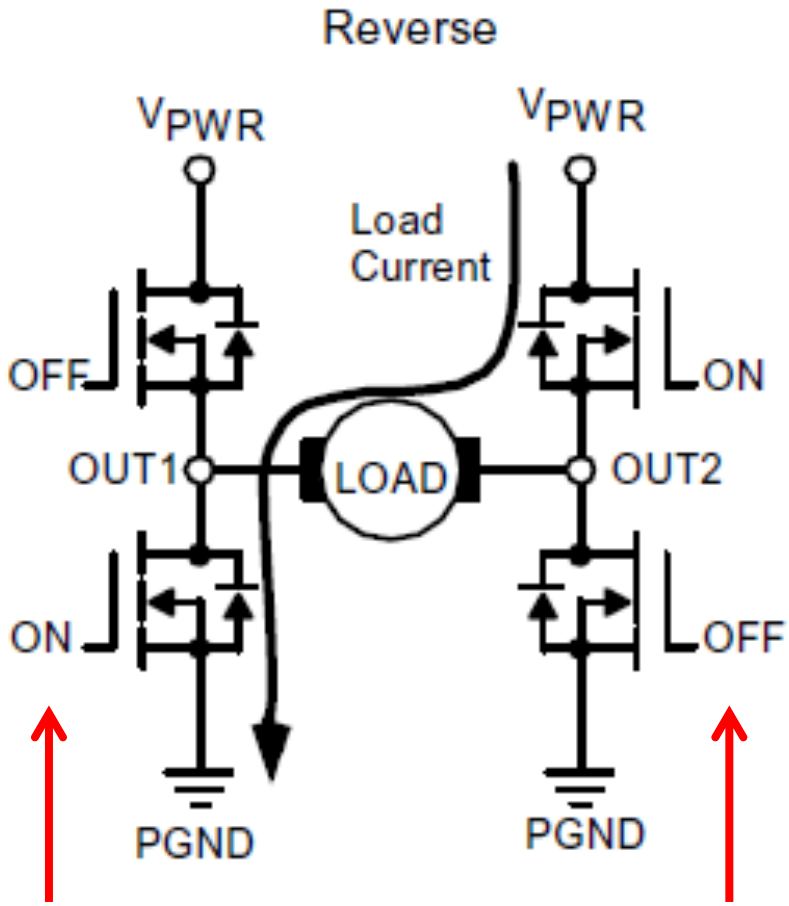


MC33887 Internal H-Bridge Block Diagram



H-Bridge DC-Motor Forward and Reverse





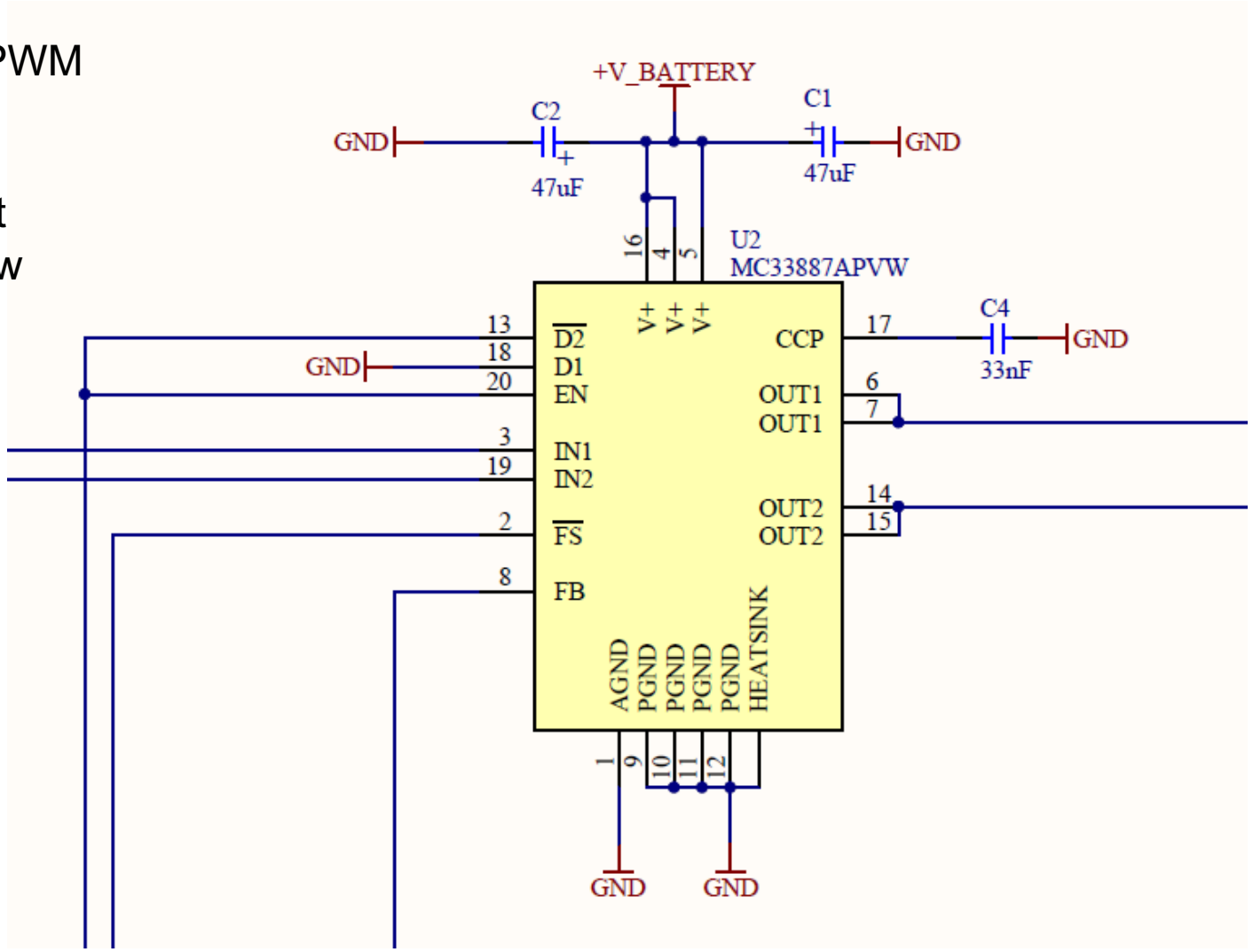
MC33887 Motor Driver

IN1: Speed via PWM

if forward,
than 100% = fast
10% = slow
0% = off

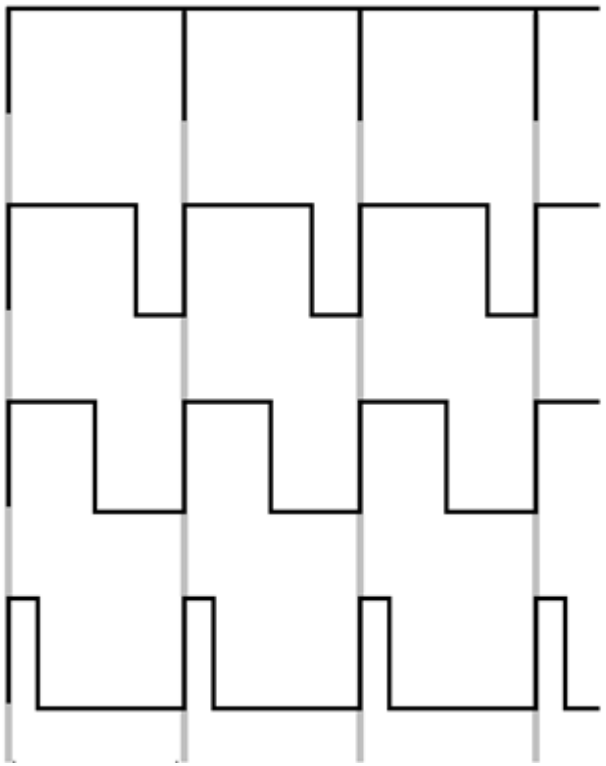
IN2: direction
via GPIO

Low: forward
High: reverse



Update Rate Main Motor Drive MC33887

Characteristic	Symbol	Min	Typ	Max	Unit
TIMING CHARACTERISTICS					
PWM Frequency (24)	f_{PWM}	–	10	–	kHz
Maximum Switching Frequency During Active Current Limiting (25)	f_{MAX}	–	–	20	kHz



10 ms update rate is fine.

But it can be much faster !

Sensors used

Sensors actually used in the simple software examples

one line scan camera

switches and potentiometers (these are not real sensors)

Sensors which can be used in more complex software

odometry sensors (reading the turns of the rear wheels)

a second line scan camera

3-axis acceleration sensor of the FRDM-KL25Z module

current sensor of the DC drive motors

optical reflection sensors

Summary of Resources used for the „Model C“ Car

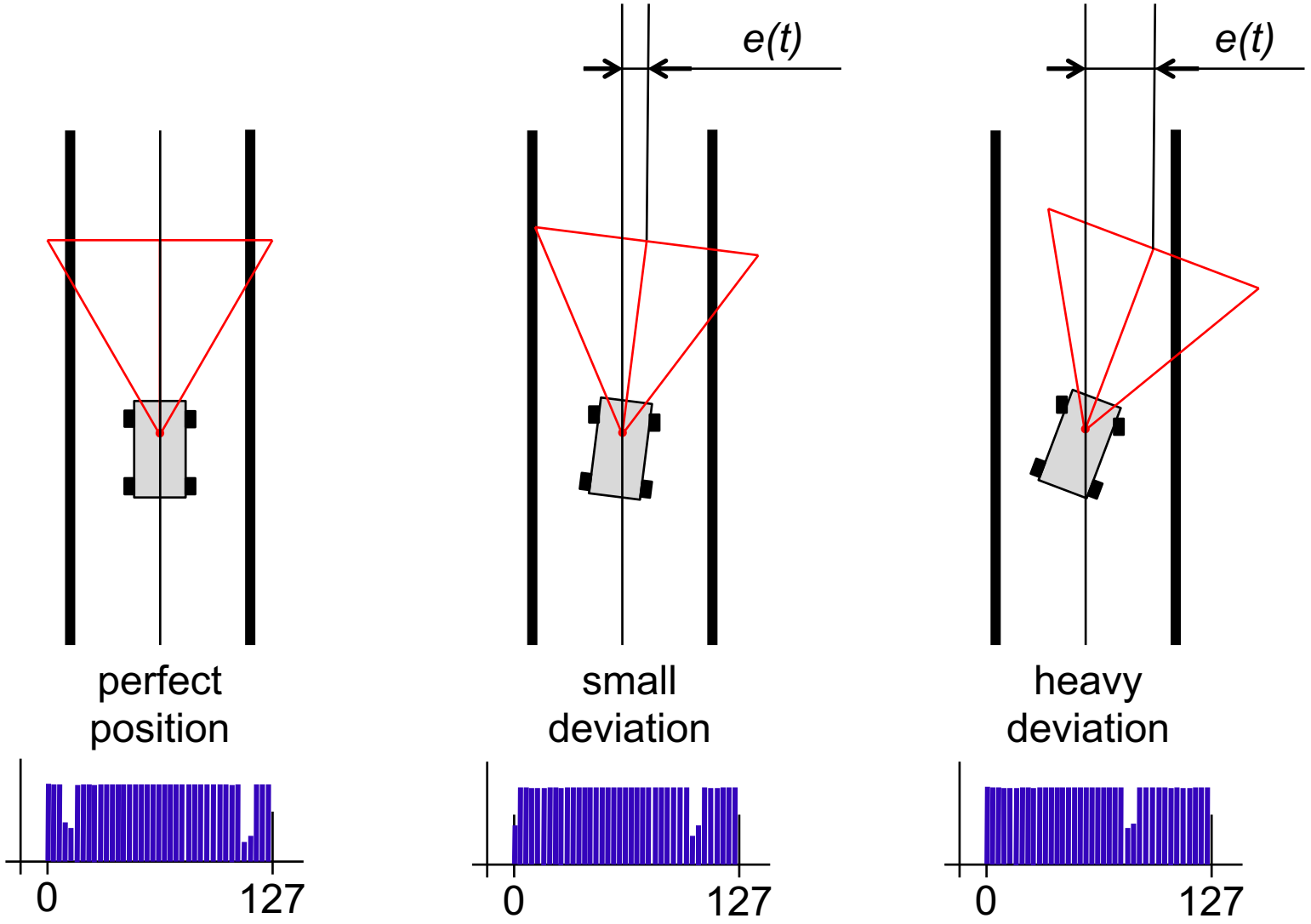
Push Button A	**PTC13**
Push Button B	PTC17
DIP Switch 1	PTE2
DIP Switch 2	PTE3
DIP Switch 3	PTE4
DIP Switch 4	PTE5
LED1	PTB8
LED2	PTB9
LED3	PTB10
LED4	PTB11
POT A	PTB3 / **ADC0_SE13**
POT B	PTB2 / **ADC0_SE12**
Speed Sensor 0	PTA1 / FTM2_CH0
Speed Sensor 1	PTA2 / FTM2_CH1

Camera Clock	**PTE1**
Camera SI	**PTD7**
Camera 0 AI	PTD5 / **ADC0_SE6b**
Camera 1 AI	PTD6 / ADC0_SE7b
Servo Out 0	PTB0 / **FTM1_CH0**
Servo Out 1	PTB1 / FTM1_CH1
Motor A IN1	PTC3 / **FTM0_CH2**
Motor A IN2	**PTC4** / FTM0_CH3
Motor B IN1	PTC1 / **FTM0_CH0**
Motor B IN2	**PTC2** / FTM0_CH1
Motor A IFB	PTE23 / ADC0_SE7a
Motor B IFB	PTE22 / ADC0_SE3
Motors Enable O	**PTE21**
Motors Fault In	PTE20
Battery Voltage	PTE29 / ADC0_SE4b

Summary of Resources used for the „Alamak" Car

Servo Motor: SM1 Signal: STERIN GPWM1		**PTA12**	TPM1_CH0
Servo Motor: SM2 Signal: STERIN GPWM2		**PTA13**	TPM1_CH1
CCD Camera:	2 CCD_AD0	**PTC2**	ADC0_SE11
	4 CCD_SI_0	**PTB8**	PTB8
	6 CCD_CLK_0	**PTB9**	PTB9
	8 + 3.3 V		
S2	Key_Value	**PTC0**	ADC0_SE14
Motor driver Infineon BTN7960 (up to 25 kHz PWM rate)			
Motor1_In1	MotorPwm1	**PTA4**	TPM0_CH1
Motor1_In2	MotorPwm2	**PTA5**	TPM0_CH2
Motor2_In1	MotorPwm3	**PTC8**	TPM0_CH4
Motor2_In2	MotorPwm4	**PTC9**	TPM0_CH5

Control Deviation – error e(t)



line scan camera data

PD Controller

To make this process quantitative, we use the quantity ‘error’, calculated in each iteration using the raw data from the camera. The error is defined to be positive in one situation and its magnitude proportional to the degree the car is off-tracked further towards the left. In another case the situation is symmetrically opposite to the first situation. This opportunity is availed by defining the error negative (differentiating from the earlier situation) and now the actuation process do exactly opposite to what happens in the first situation. Each of two try to approach towards the situation where car is on track and so error is defined to be numerically zero, which signifies no actuation is needed.

Once the error is obtained, we calculate a ‘correction’ term which is fed to the actuator. The actuator turns the servo motor (and hence the wheels of the car) clockwise or anticlockwise depending on the sign of error. The degree of turn is estimated using the magnitude of the error term which is not merely proportional but is obtained using a **PD controller**, which has the following general form.

Correction = KP × error + KD × delta(error)

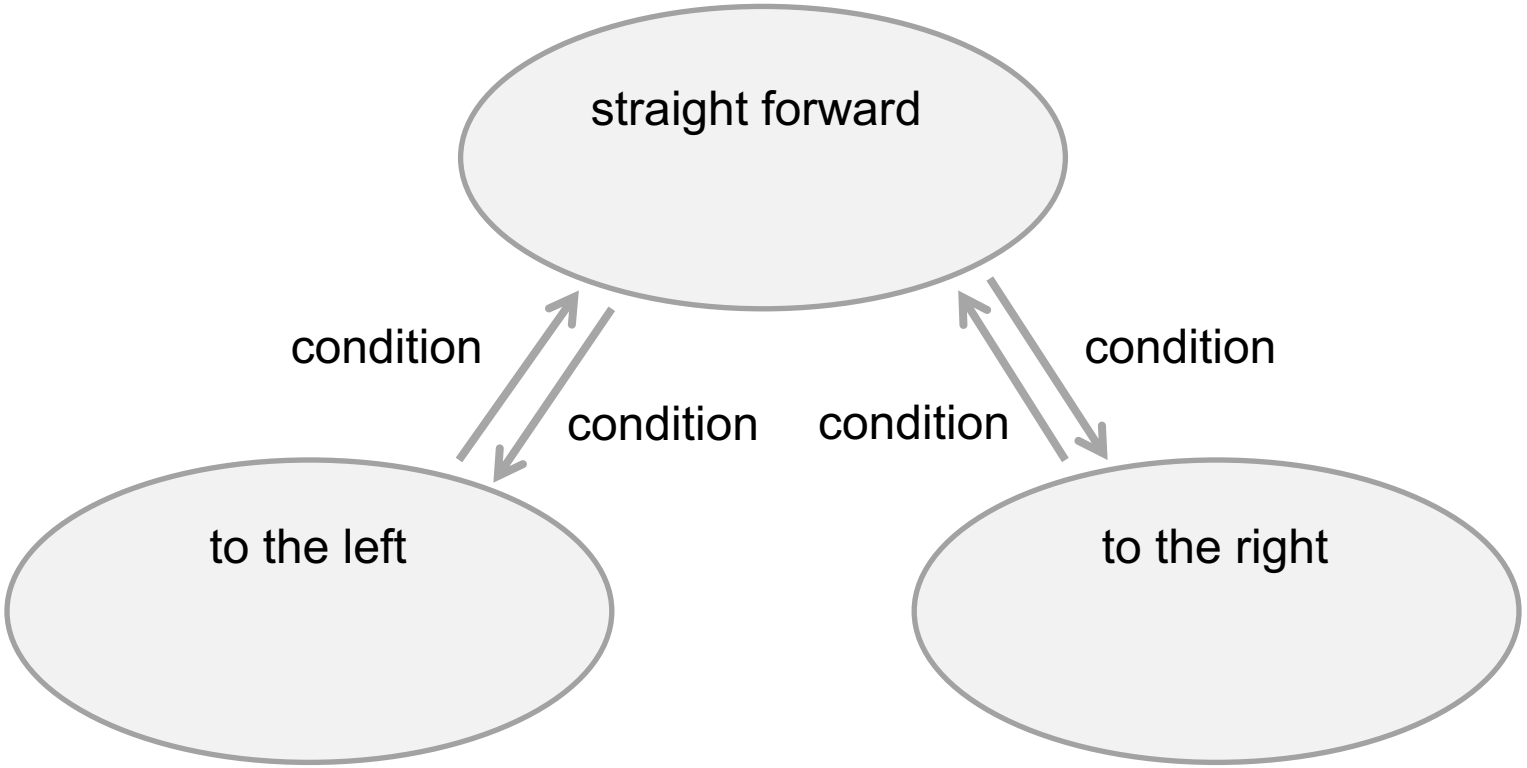
Improvement Options for Control Servo Settings

Plausibility check:
When setting the servo for direction control, do a plausibility check for Direction: if the direction from one step to the other changes significantly, probably there has been a mistake or lost signal.

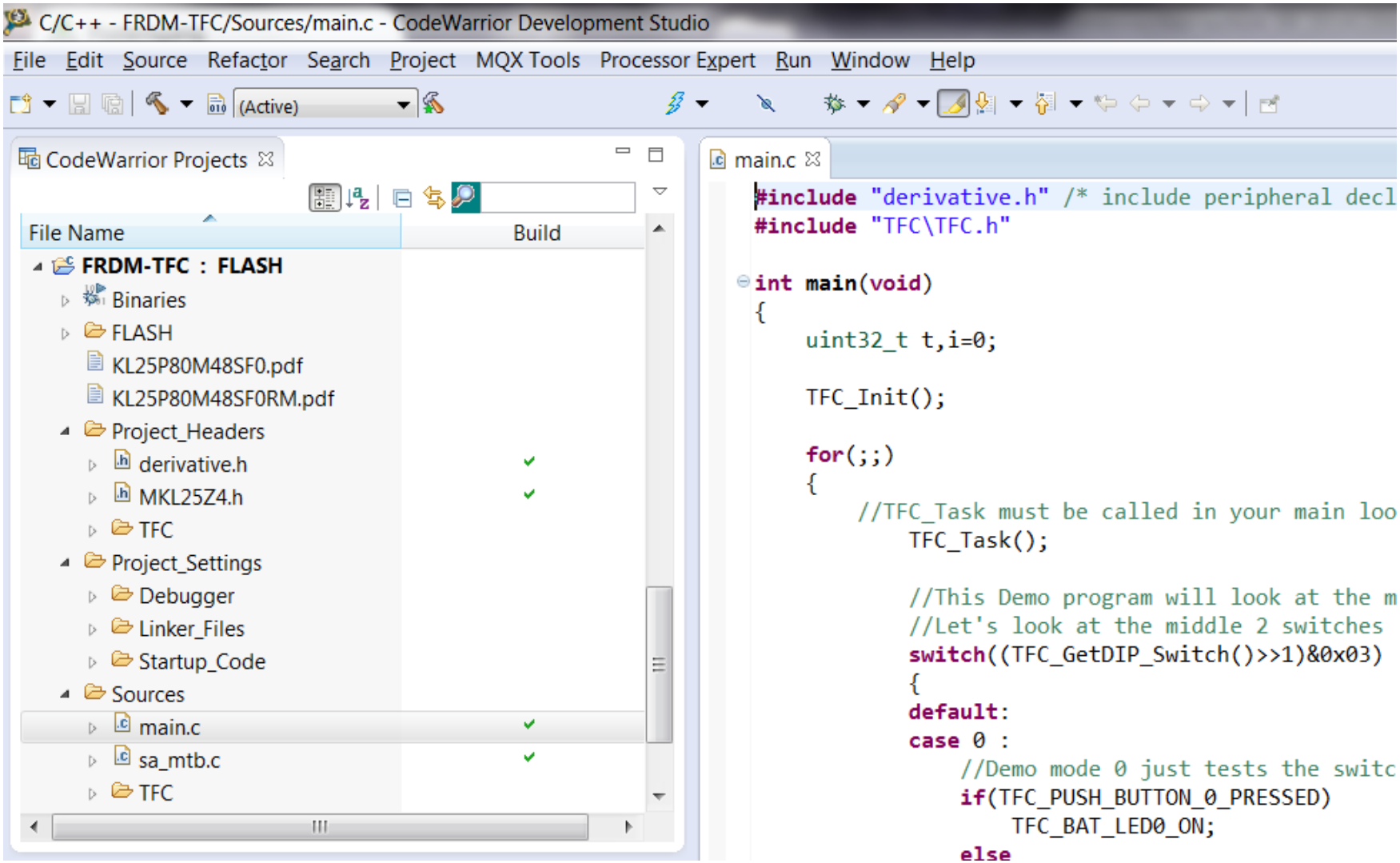
Take line samples more often:
Try to speed up the line image capturing process and take more line samples, this could result in better results of the calculation.

Find line position and line width:
The actual sample code is just looking for the minimum im the array.
Try to find maximum in the array and miminum in the array, then define a threshold value, so you can better distinguish between black and white.

State Machine



Example Project: FRDM-TFC (FRDM-TFC.zip)



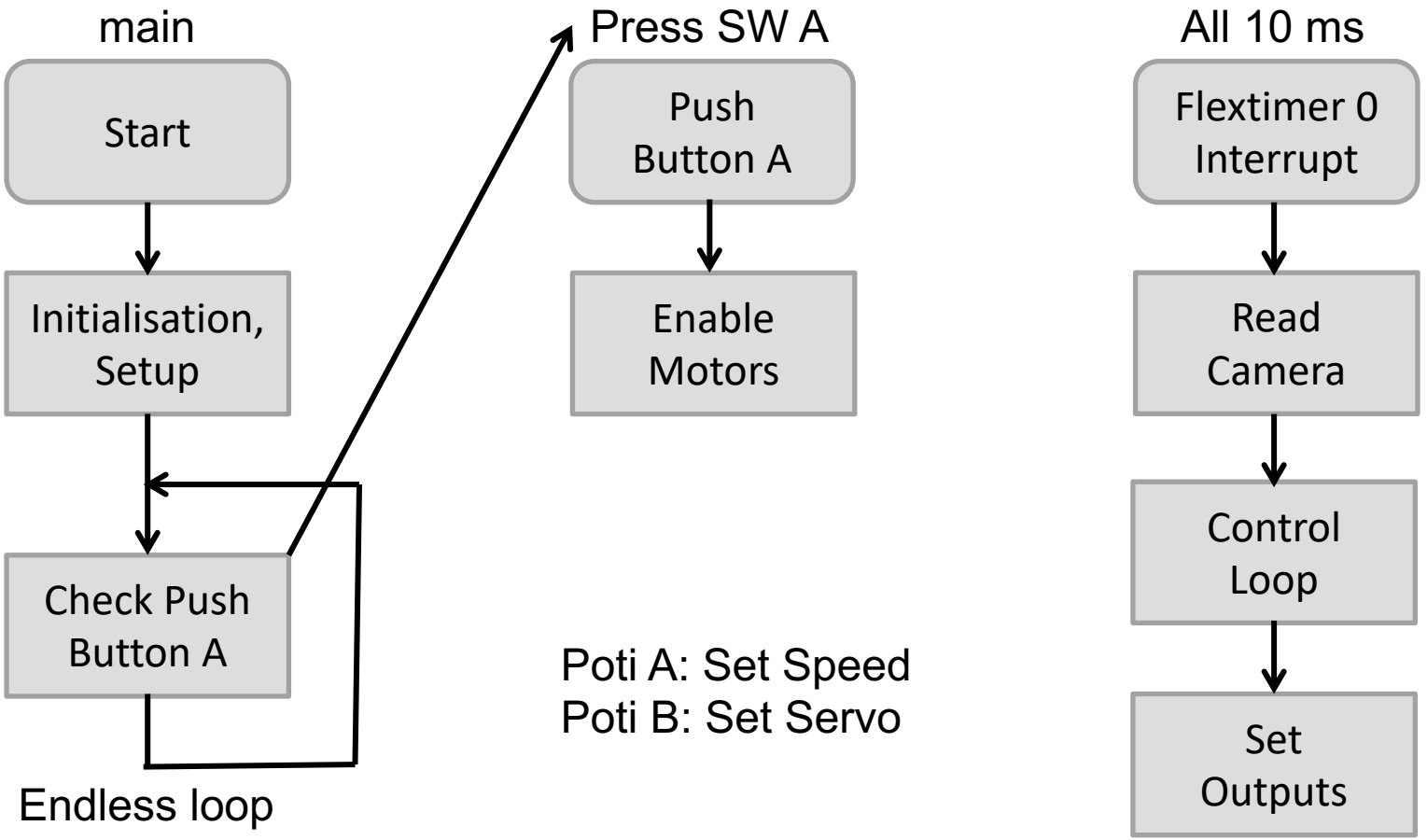
Interrupts used in FRDM-TFC Software

PIT_IRQHandler() in TFC_ADC.c	Trigger ADC in regular time intervals
ADC0_IRQHandler() in TFC_ADC.c	Delay and ADC conversions
SysTick_Handler() in TFC_ARM_SysTic.c	Delay Timer Ticker
FTM1_IRQHandler() in TFC_Servo.c	Count up Servo Tick Variable
UART0_IRQHandler() in TFC_UART.c	Sending Out data via UART

FRDM-TFC is an example software from Freescale. You can analyze it and use it as a starting point for your own software.

FRDM-KL25Z NXP Cup Car Basic Software Structure

Minimalistic example project for FRDM-KL25Z (NXP_Cup_Minimal.zip)



Initialisation, Setup, Configurationen

Configure clock to 48 MHz

Switch on all clocks (GPIOs, FlexTimer0, ADC, UART3)

Configure GPIOs

Configure clock and timeout for TPM

Configure pins for TPM

Configure TPM channels

Set FlexTimer0 Modulo

Configure AD Converter

Enable interrupts

Endless Loop

Wait for SW A to be pressed

Flow of FTM0 ISR (triggered every 10 ms)

Operation	Result
clear FTM0 ISR flag	
capture potA and potB	
capture LineScan image	array of 128 bytes
find the black line on the left and right side	
find the middle of the road	
plausibility check / state machine	
PD control algorithm	
set servo position	PWM value
set drive motor speed	PWM value
identify start / stop	flag

Option: Differential Speed Control

If you go to a curve, identify if it is a right curve or a left curve.

Brake more rapidly with the inner wheel of the curve.

What about the Light? We did not talk about the Light!

Light (illumination) is not strictly necessary for NXP Cup cars.
However, it is recommended to have a good working illumination on the car, which can be switched on or off. And it is recommended to have software options available which are optimized to work with and without illumination. Experience shows, that the conditions may vary a lot and the performance of the car depends very much on the lighting at the track. So for better flexibility it is good to have lights on the car available.

General Recommendations for Software Development

Use example code:
- software examples from the microcontroller manufacturer
- code ideas from your customers

Go step by step in your project:
- develop debug and test cases
- go further based on known good results
- document your changes and save all intermediate results!

Use the Internet:
- The NXP Cup Knowledge Center
https://community.nxp.com/docs/DOC-1284
- Videos on YouTube (search for: Freescale Cup, NXP Cup)

Ask questions!

Fouls, Failure and Disqualifications

The rules will be interpreted by NXP and the organizing committee of the event.

Foul, is a minor infraction, which results in a time penalty.

Failure, results in the current attempt time not being recorded. Subsequent attempts are allowed.

Disqualification is a major infraction which results in all times not being recorded.

A referee will determine whether the racing car ran out of the race track and assign time penalties.

No part of the car shall exceed dimensions of 250mm/9.85in (W) x 400mm/15.75in (L) x 305mm/12in (H).

Start/Stop Line

The car must be able to automatically stop within two meters of the starting line after finishing the race.

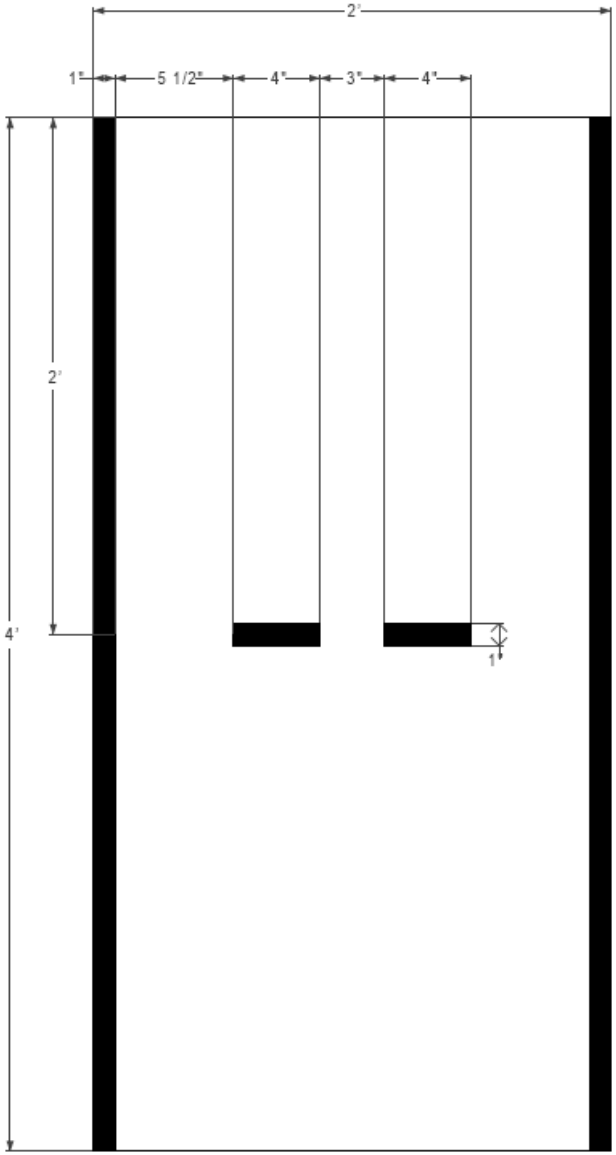
Any of th
consider
penalty a

You should have different software options with and without stop line recognition.

1. The racecar to leave the starting area within 3 econds after beginning of the race [+ second].

2. **The racecar fails to stop 2 meters / 6 feet or leaves the track after crossing the finish line [+1 second].**

3. The racecar exits the racetrack after crossing the finish line [+1 second]



Failure and Disqualification

Any of the following conditions will be considered a ***failure*** and no time will be given:

1. Three or more wheels leave the race surface

2. The racing te fails to get prepared for the attempt within the two (2) minutes allotm

3. The t ecar after the technical inspect

4. The in 120 seconds after leaving the star

5. The team member touches the car at any time between start and finish as "Start" is once part of the racecar crosses or partially crosses the starting line and "Finish" once the vehicle crosses the finish line.

Two wheels can leave the track, so you can drift around the corners to maximize speed.

Any of the following conditions will be considered a ***disqualification***:

1. Any off track equipment or behavior that may influence or impede cars
2. Doing a Disallowed Modification any time after the technical inspection
3. More than one team member in the race field
4. Any cheating during the competition
5. Failure to pass the technical inspection

Option: Use Different Strategies

When your team is called, one (1) team member may remove the racecar from inspection area and has two (2) minutes to prepare the car. These following actions are allowed during the preparation time:

1. Configure parameters via on-board interfaces. (Switches, Knobs, etc.)
2. Alter the angle of your camera
3. Change batteries

It is not allowed to connect the racecar to any computer devices to upload, reconfigure or change programming on the racecar.

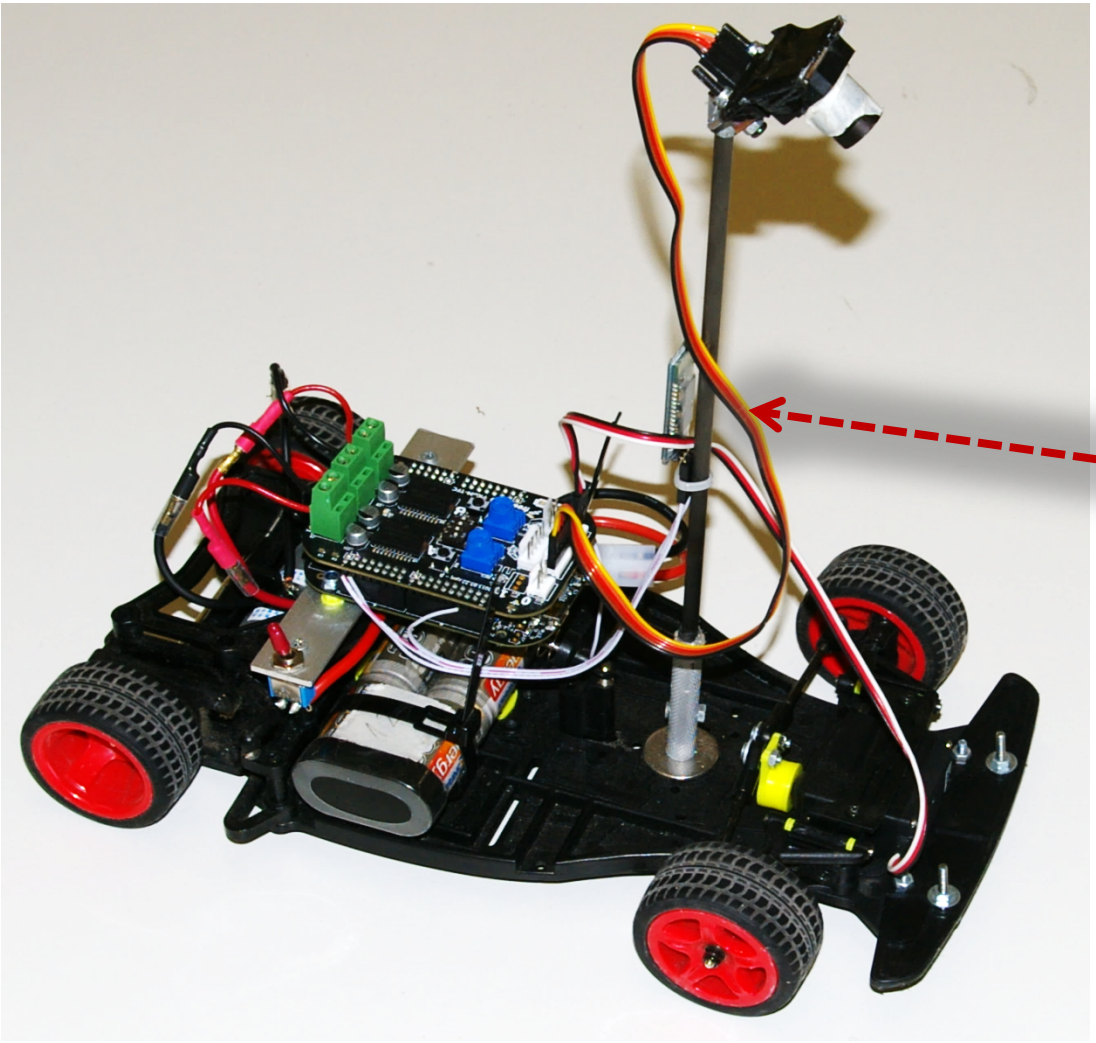
Before the car starts, the team must signal "Ready" to the referee before starting car.

After the referee confirms "Ready", the vehicle should leave the starting area within 30 seconds.

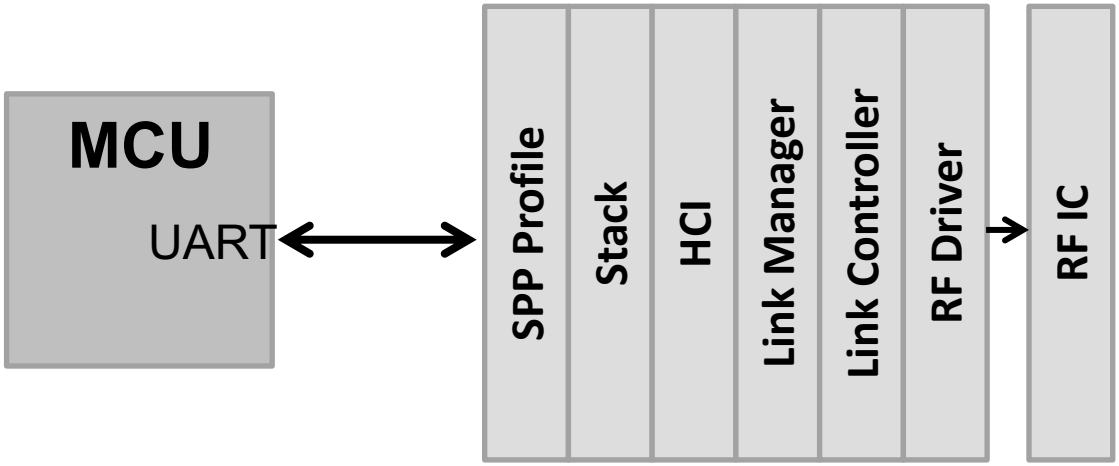
Store different strategies at switch positions SW1, SW2 …

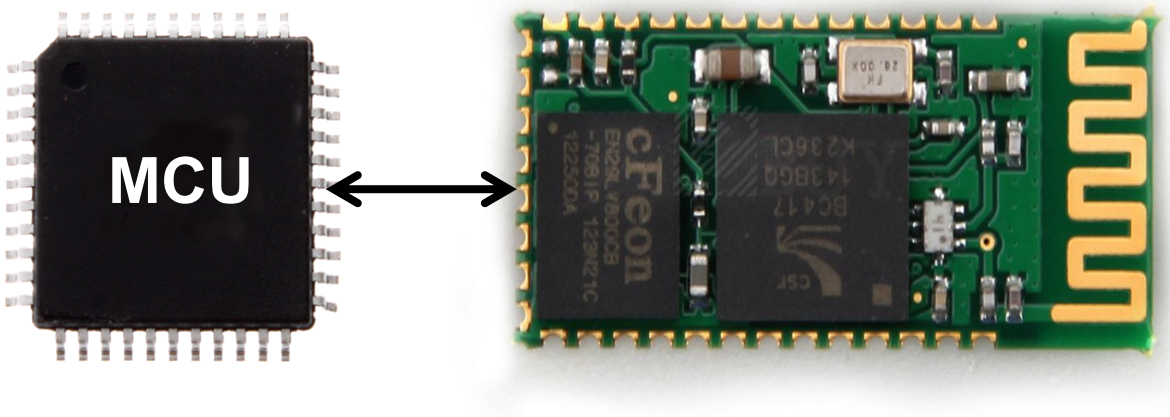
The team has three (3) attempts to complete one (1) lap.
The first (not the best) completed time will be recorded.

Bluetooth Wireless Telemetry (NOT during the race)

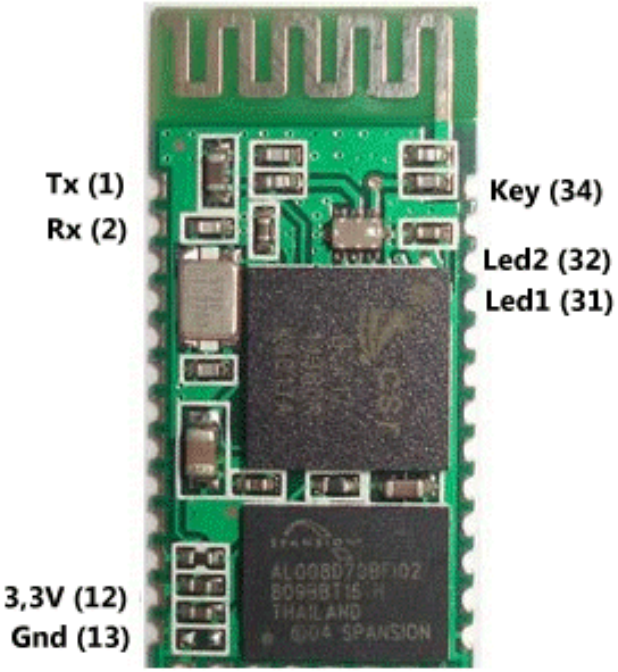


Bluetooth Serial Port Profile

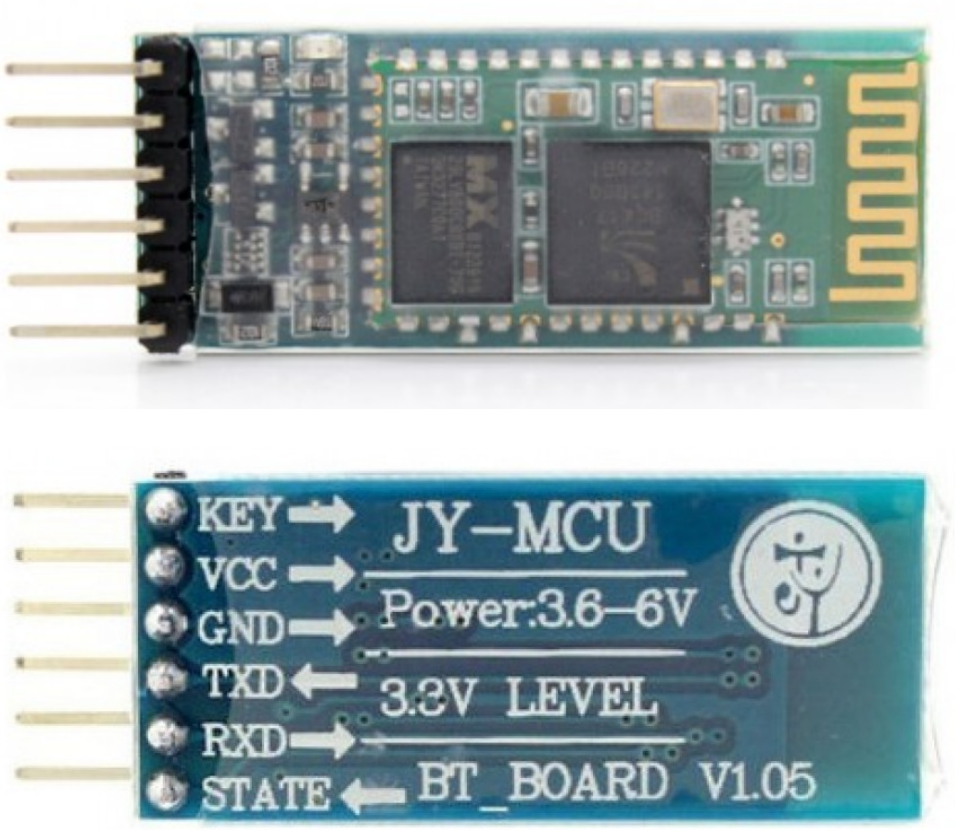




HC-05 Standard Bluetooth Module



HC-05 Bluetooth Module



Bluetooth Module auf Trägerplatine

Bluetooth Node Roles: Master and Slave

Master: has a name and a MAC address

can scan and discover slave nodes

can be paired to one or several slave nodes

can provide a passkey for pairing with slave node

can send data to slave nodes and receive data from slave nodes

Slave: has a name and a MAC address

can be discovered by master node

can be made none-discoverable

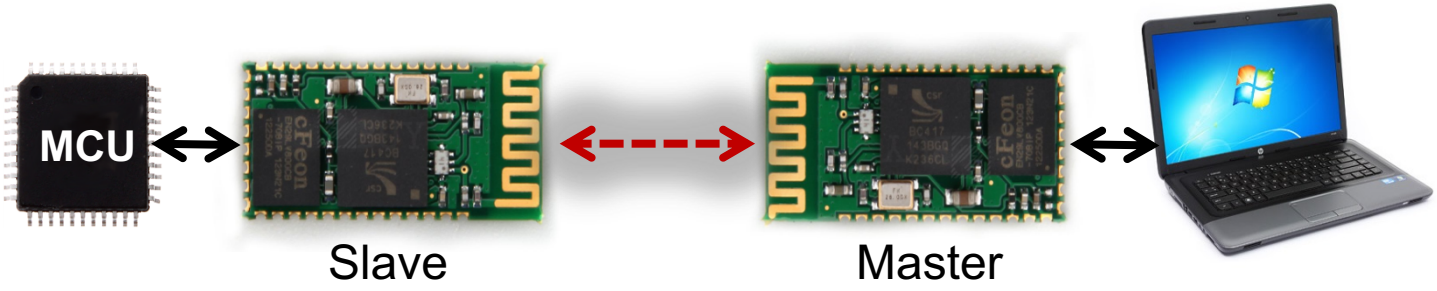
can be paired to just one master node

can request a passkey from the master node

can send data to master node and receive data from master node

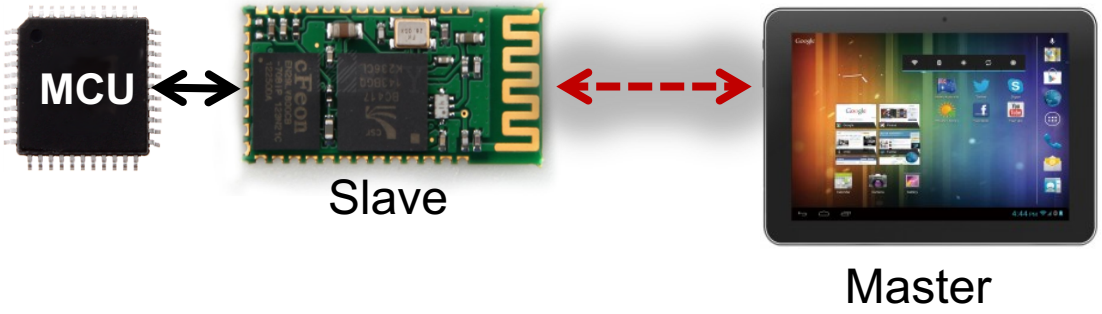
HC-05 Module Serial Port Profile Connectivity Options

a) To another HC05 module



b) To a standard PC

c) To a tablet computer with SPP



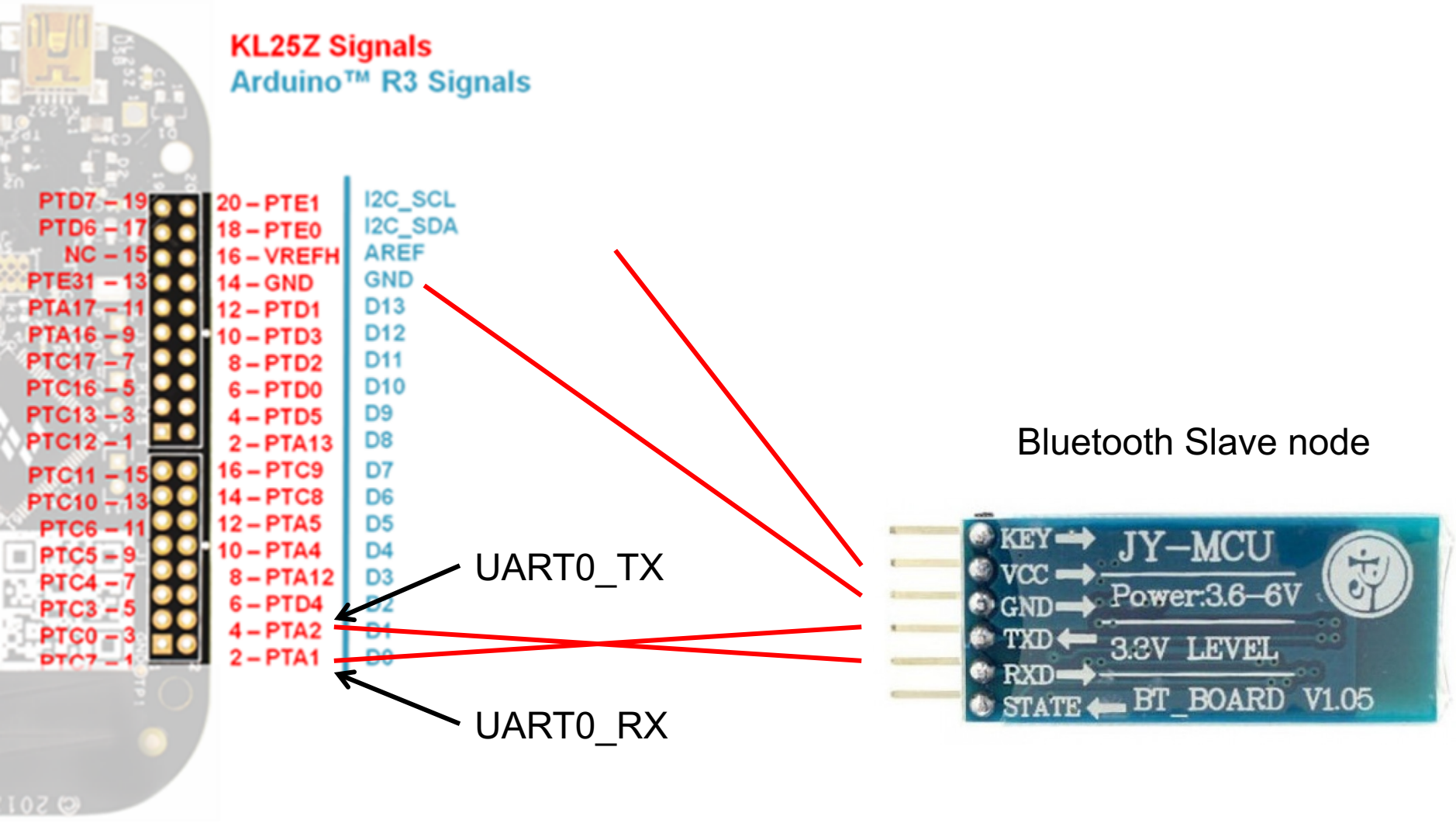
HC-05 Bluetooth Module Configuration

Pin 1: Module UART Tx (Connect to MCU UART Rx)
Pin 2: Module UART Rx (Connect to MCU UART Tx)
Pin 12: 3,3 V Supply Voltage
Pin 13: Ground
Pin 31: LED1 indicator of Bluetooth status
Pin 34: Key input:
H – AT command mode for configuration
L or n.c. – data communication mode

Default mode:	data communication mode
Default Bluetooth role:	slave role
Default data baudrate:	9600, 8N1
Default name:	HC-05
Default pairing key:	1234

New name:	**HC-05-XXX**
New data baudrate:	**115200, 8N1**

HC-05 Module Connection to FRDM-KL25Z

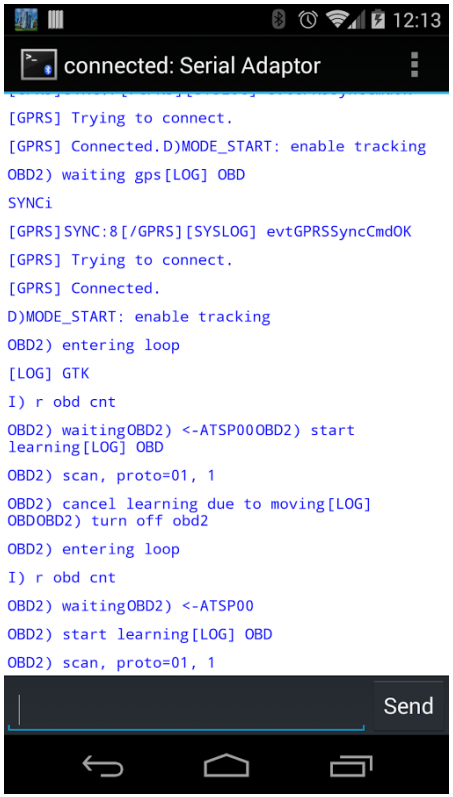


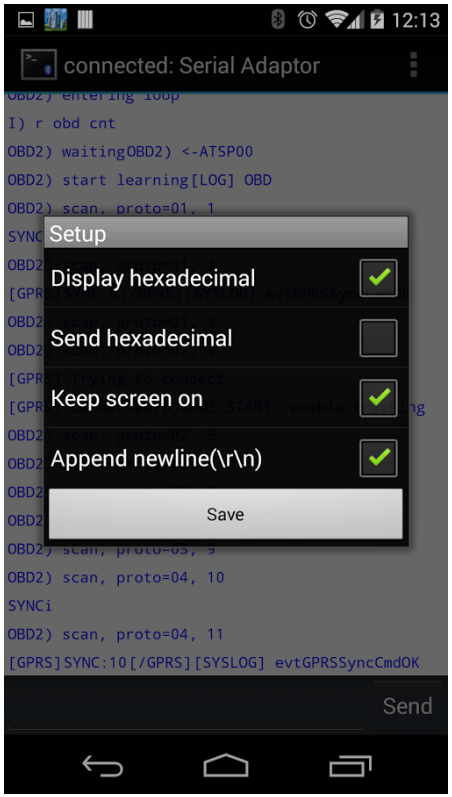
Bluetooth Terminal Android App

kostenfreie Android App

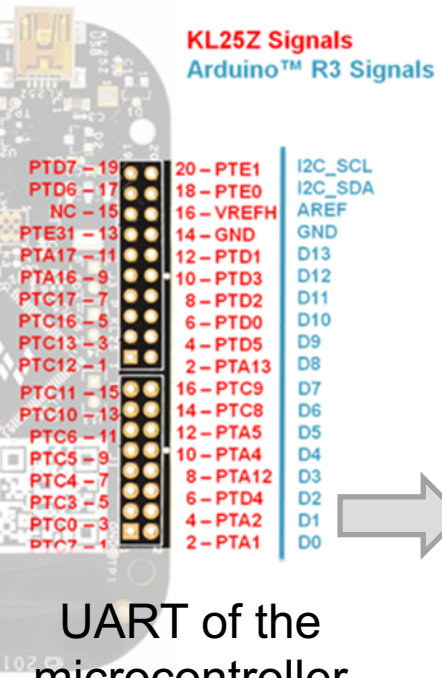
Funktion:
Empfangen und Anzeigen sowie Senden von Daten über Bluetooth

Erforderliche Android-Version:
4.0 oder höher



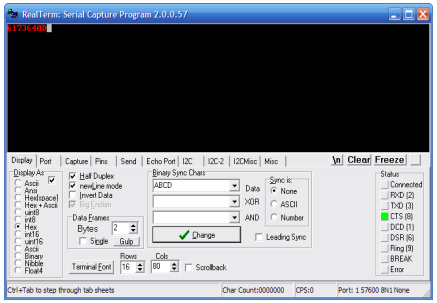


Adjustment of Communication Speed



UART of the microcontroller 115200, 8N1

Bluetooth Module 115200, 8N1



Terminal Program 115200, 8N1

TECHNISCHE HOCHSCHULE DEGGENDORF

Technische Hochschule Deggendorf – Dieter-Görlitz-Platz 1 – 94469 Deggendorf