Was unterscheidet Microcontroller von einem PC (Laptop) oder einem Handy

An einen uC kann man eine Menge Sensoren und Aktuatoren anschließen.

Mit den Sensoren kann der uC Werte aus der physikalischen Umwelt erfassen.

Mit den Aktuatoren kann der uC Motoren, Servos Relais …. ansteuern.

Der uC kann im Gegensatz zu PC, Handy,Raspbery auf Sensoränderungen extrem schnell reagieren und neue Werte auf die Aktuatoren ausgeben.

Beispiele für anspruchsvolle uC Anwendungen:

- Steuerung und Regelung eines Quadrocopters der uC hält mithilfe von Lagesensoren und durch Drehzahlregelung an den Motoren den Quadrocopter im Gleichgewicht

Welche IO-Pins des uC sind beim Liniensuchenden Roboter mit welchen Sensoren (Aktuatoren) verbunden:

6x Linensensor => 6x AnalogIn am uC

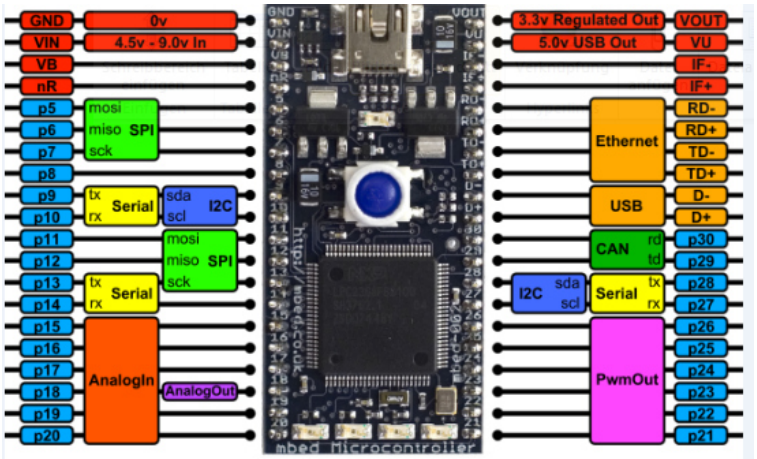
Motor 2x:
- 1x PWM-Out
- 2x Digital -Out

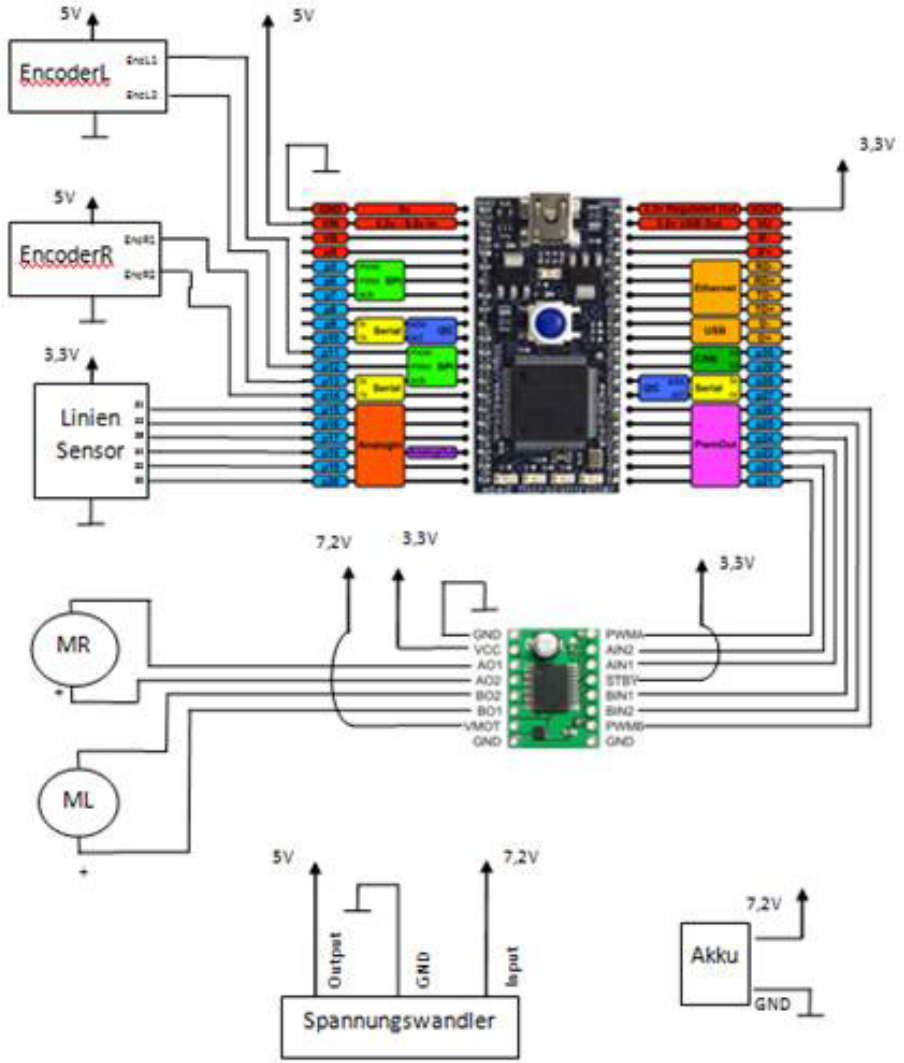
Kommunikation mit PC über Bluetooth => 1x Serielle Schnittstelle

Kommunikation mit PC über USB => 1x Serielle Schnittstelle

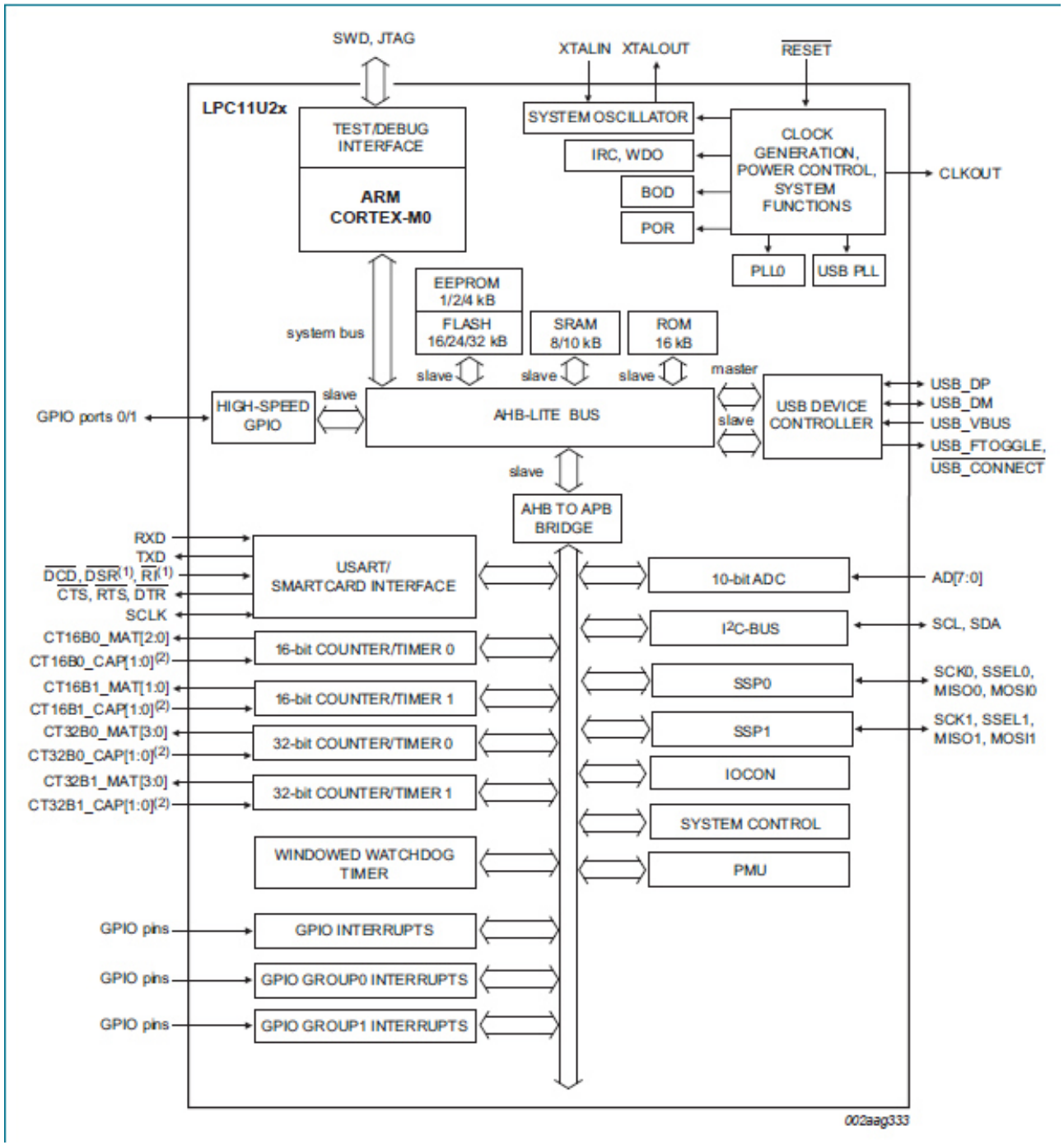
WegEncoder: 2x 2 Digital In

Was unterscheidet Microcontroller von einem PC (Laptop) oder einem Handy





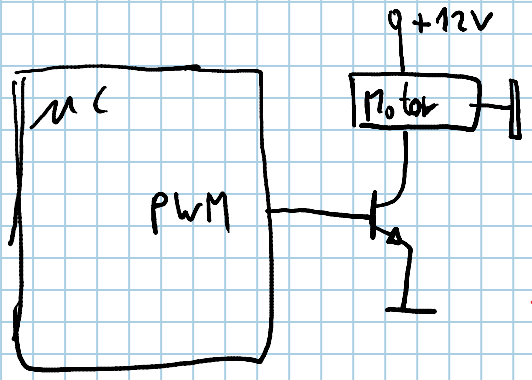
Das Innenleben des Microcontrollers

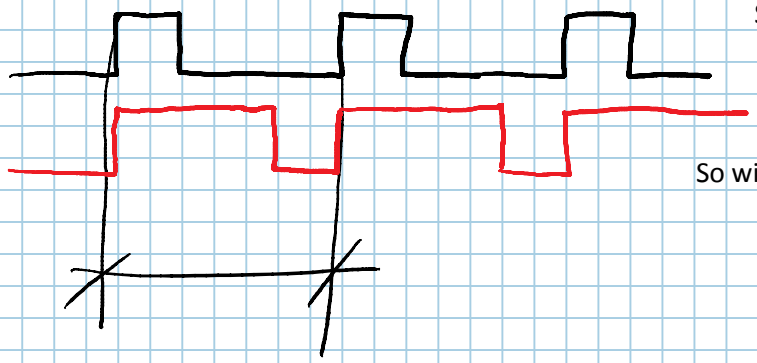


S2

Pulsweitenmodulation

Motor wird während einer Periode verschieden lange eingeschaltet



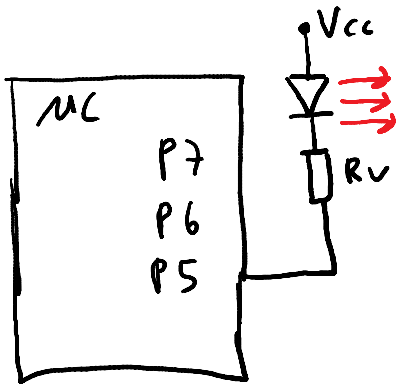


So wird die PWM bei wenig Leistung gepulst

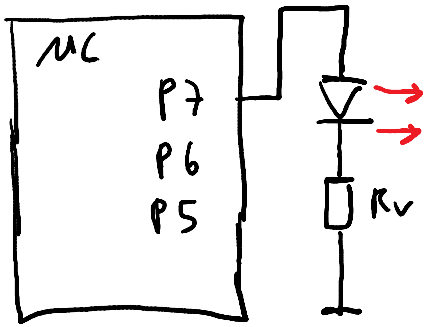
So wird die PWM bei viel Leistung gepulst

Eine PWM Periode

LED'S mit DigitalOUT ansteuern



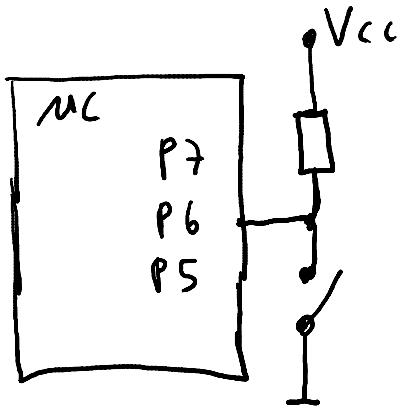
LED wird mit P5=0 eingeschaltet und mit
P5=1 ausgeschaltet



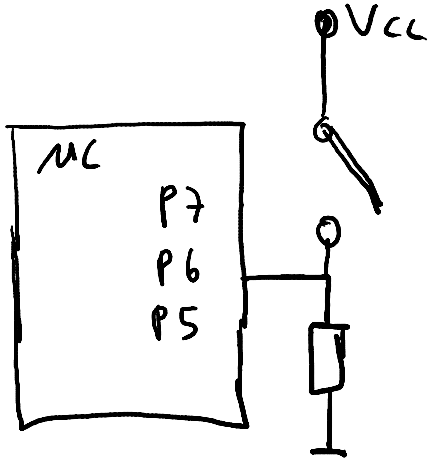
LED wird mit P7=1 eingeschaltet und mit
P7=0 ausgeschaltet

Man kann ein Port entweder als Eingang oder als Ausgang schalten aber **nicht!! beides**

Schalter mit DigitalIn einlesen



Bei geschlossenem Schalter ist P6==0
Bei geöffnetem Schalter ist P6==1



Bei geschlossenem Schalter ist P6==1
Bei geöffnetem Schalter ist P6==0

Bitoperationen

```
// Der Bitschiebe-Befehl in C/C++

int  a;

a = B#0001;  // in a steht Binär 0001

a = a << 1;  // in a steht jetzt B#0010
a = a << 1;  // in a steht jetzt B#0100

a = a >> 2;  // in a steht jetzt B#0001
a = a >> 1;  // in a steht jetzt B#0000

a = B#000011;
a = a << 1;  // in a steht jetzt B#000110
```

Bitwise OR

1000 | 0001 = 1001

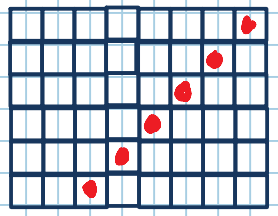
Bits nach links schieben und 1en nachschieben

```
a = 00001;
a = a << 1;  // 00010

// ich hätte aber gerne das 1en nachgeschoben werden
// wie könnte man das machen ?

a = (a << 1) | 00001; // 00011
a = (a << 1) | 00001; // 00111
```

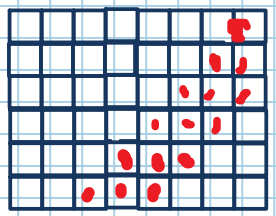
S5

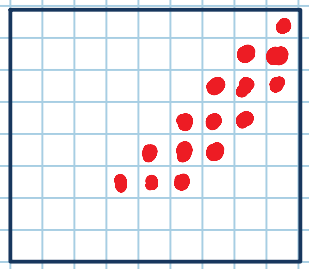
RunLights

1er Lauflicht links

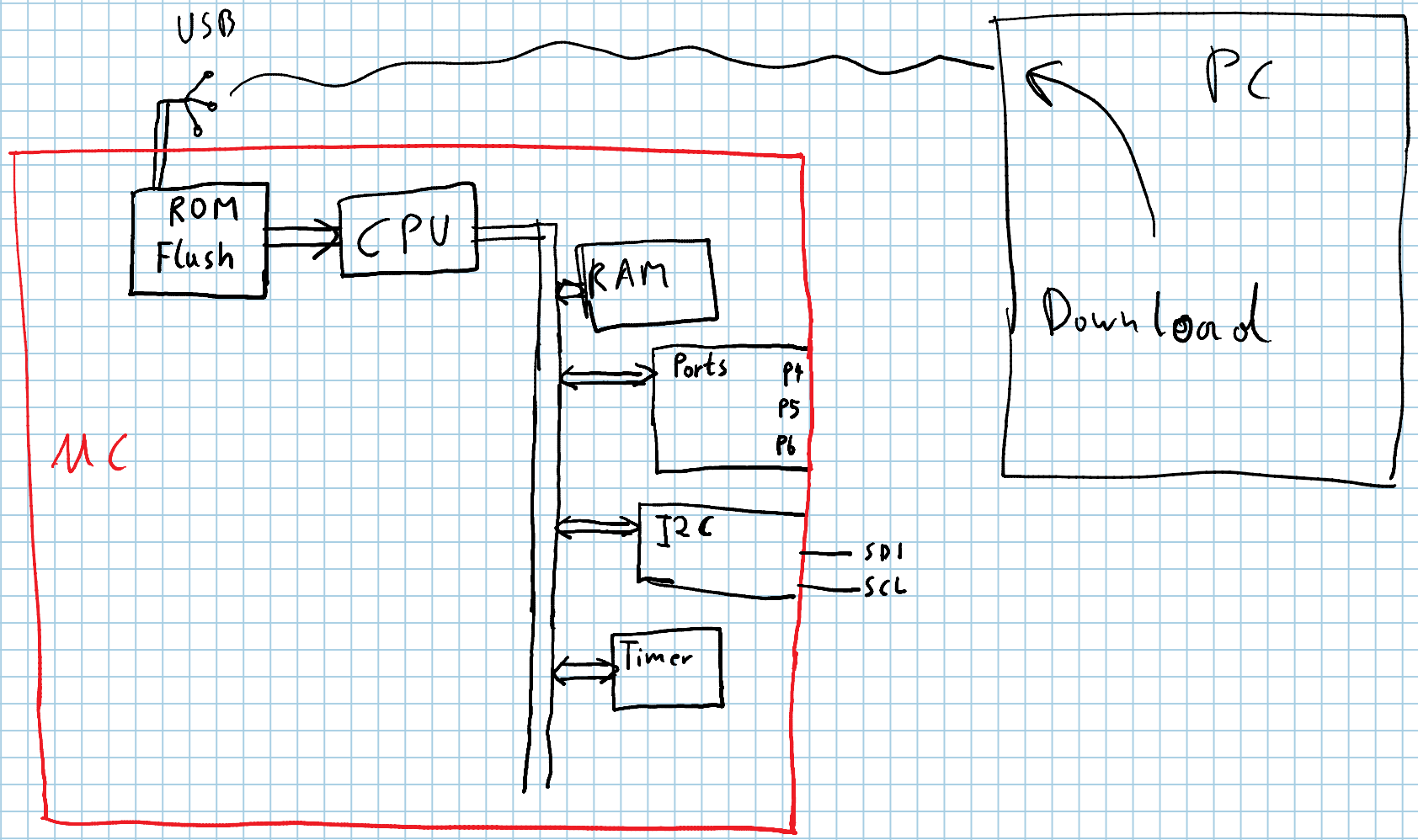


3er Lauflicht links





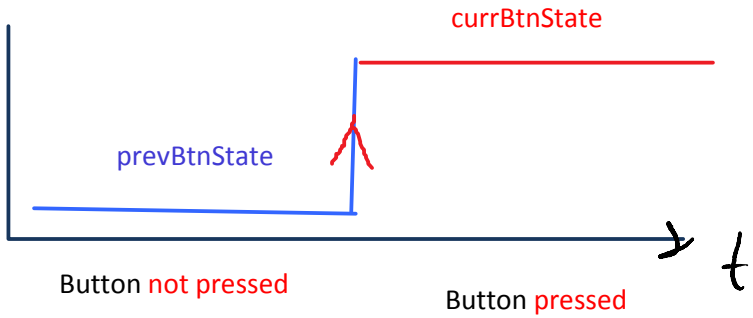
S23


USB
PC
Download
ROM
Flush
CPU
RAM
Ports
P4
P5
P6
µC
I2C
SDI
SCL
Timer

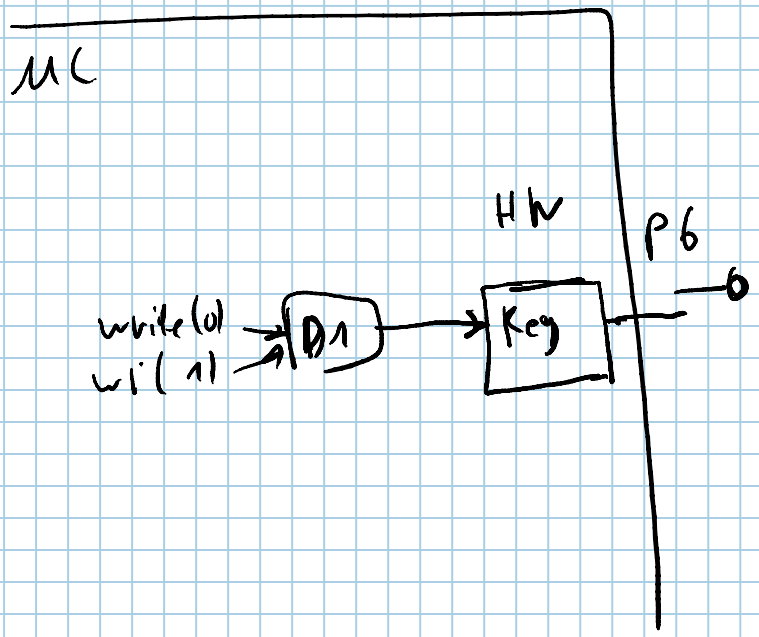
Lauflicht über Taste (Button) weiterschalten

Das Lauflicht soll sich bei jedem **Button-Click** also einem
Zustandswechsel der Taste von 0 auf 1 um einen Schritt weiterbewegen.

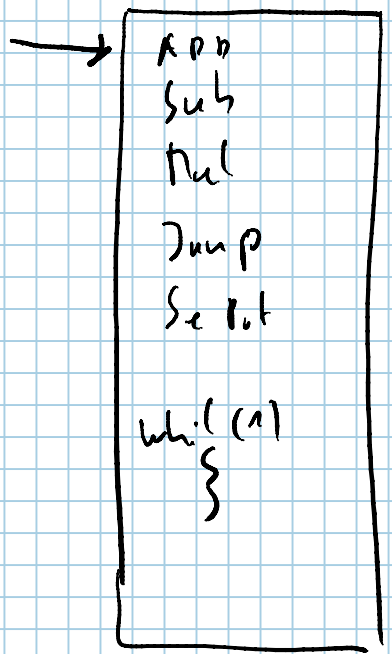
Um das zu erreichen muss unsere Tastenabfrage in der Lage sein den
Zustandswechsel der Taste von 0 auf 1 zu erkennen.



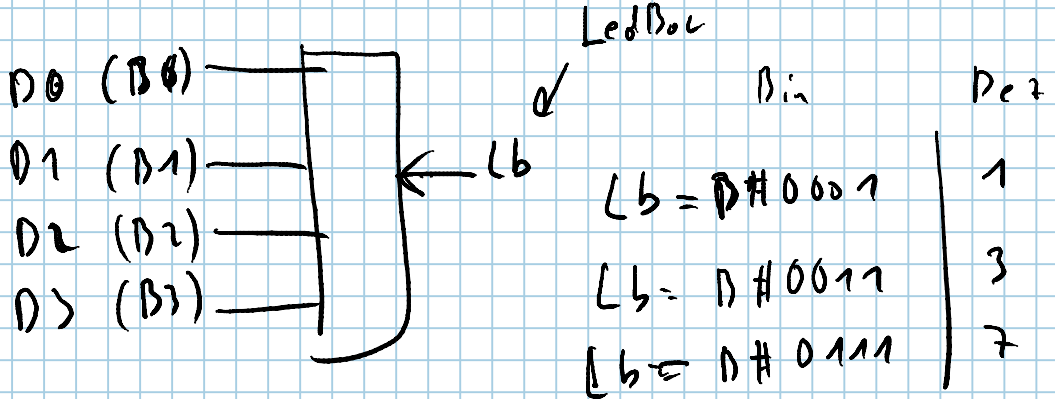
Wir werden eine Abfragefunktion CheckButton() entwerfen
Welche 1 zurückliefert wenn eine steigende Flanke erkannt wurde



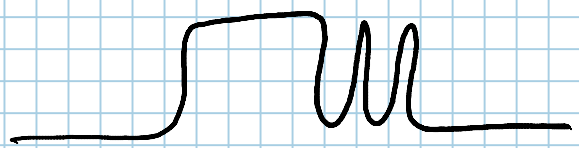
Program Memory

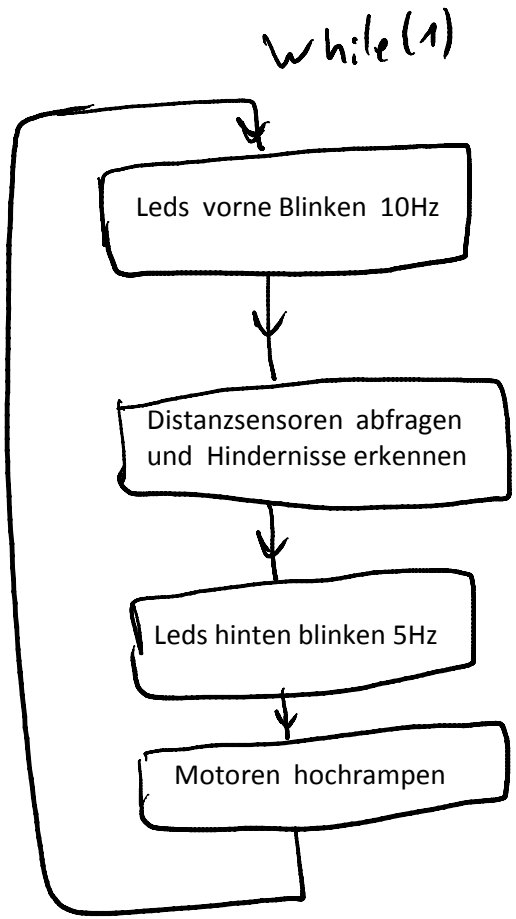


Bits zu einer Bitgruppe zusammenfassen

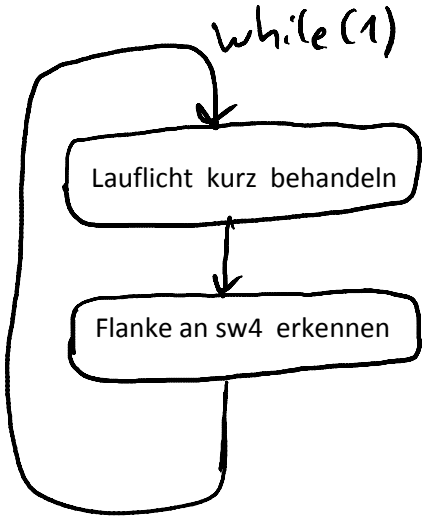


Kontaktprellen an der fallenden Flanke



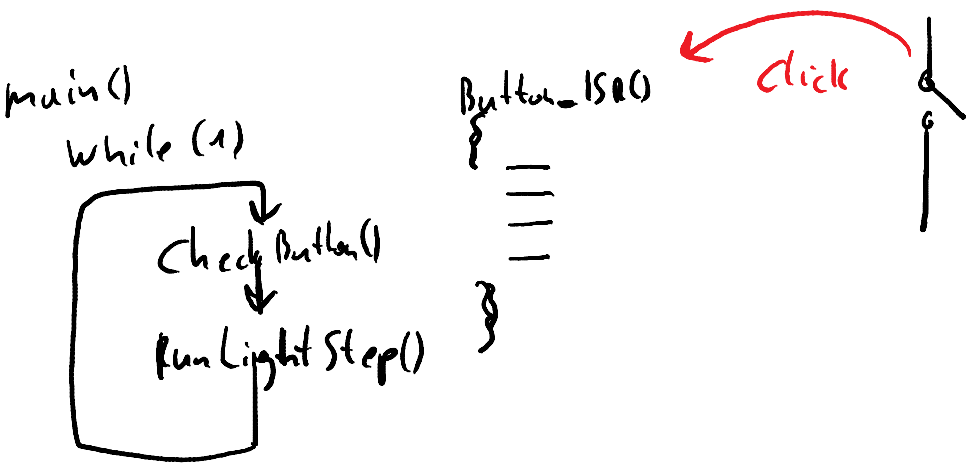


Der uC ruft in der Hauptschleife alle Tasks die er bearbeiten muss auf.



Die Lösung mit Interrupts

Interrupts sind Funktionen die von HW-Ereignissen aufgerufen werden
unabhängig davon was der uC im main() gerade tut.



Die while(1) Schleife im main() läuft vor sich hin
und sobald der Button eine steigende Flanke hat
wird von der **uC-Hardware** main() **unterbrochen!!!**
und Button_ISR() aufgerufen

Deswegen heisst dieser Mechanismus auch **Interrupt**
Weil durch den **Interrupt** das main() **unterbrochen** wird.

Wenn Button_ISR() beendet ist (return aufgerufen wurde) ,
arbeitet main() dort weiter wo es unterbrochen wurde.