Univerza v Ljubljani
Fakulteta za elektrotehniko

Žiga Mahne
(64050265)

Števec prestav na motociklu

Seminarska naloga

pri predmetu
Elektronska vezja

V Cerknici, 21. rožnik 2011

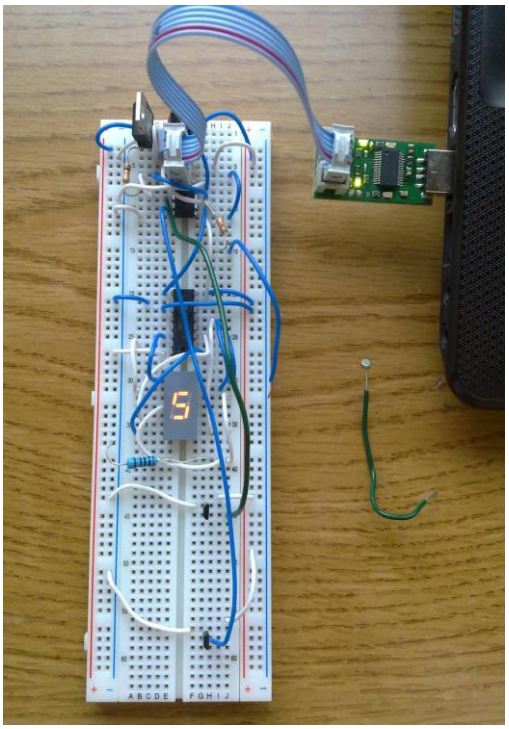
KAZALO:

1. UVOD

Motoristi pogosto niso točno prepričani v kateri prestavi je motocikel, kar ni dobro na primer pri prehitevanju, speljevanju, semaforjih... Zato sem si pri predmetu Elektronska vezja zamislil za projektno nalogo števec prestav.

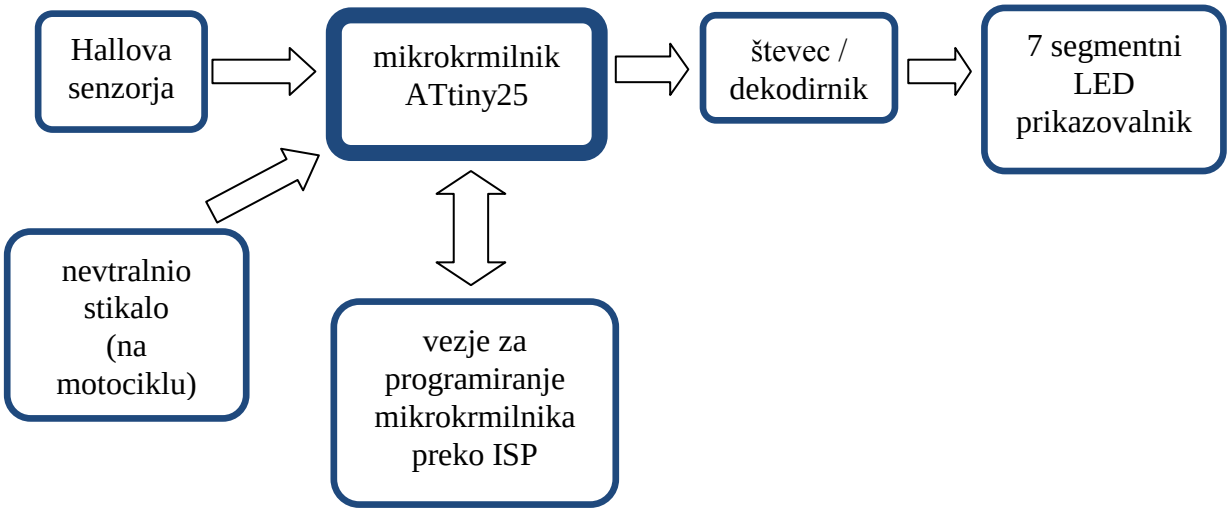
Na spletni strani [1] sem zasledil realizacijo, katera mi je bila primerna po težavnosti ter hkrati zanimiva, ker ustreza vsem vrstam motociklov. Namreč novejši motocikli imajo predpripravljena signala o trenutni hitrosti in trenutnih vrtljajih agregata, preko katerih števci prestav določajo trenutno prestavo. Realiziran števec prestav je univerzalen, saj se prestave preračunavajo preko dveh Hallovih senzorjev, katera sta nameščena na okvir motocikla tik pri ročici menjalnika, na ročici pa je majhen magnet, kateri proži senzorja.

Srce vezja je mikrokrmilnik Atmel ATtiny25 iz družine tinyAVR, kateri so najpreprostejši in enostavni za učenje in spoznavanje mikrokrmilniških vezij. Mikrokrmilnik sem sprogramiral s programsko opremo AVR studio preko USB programatorja, katerega so mi prijazno posodili v laboratoriju LPVO.

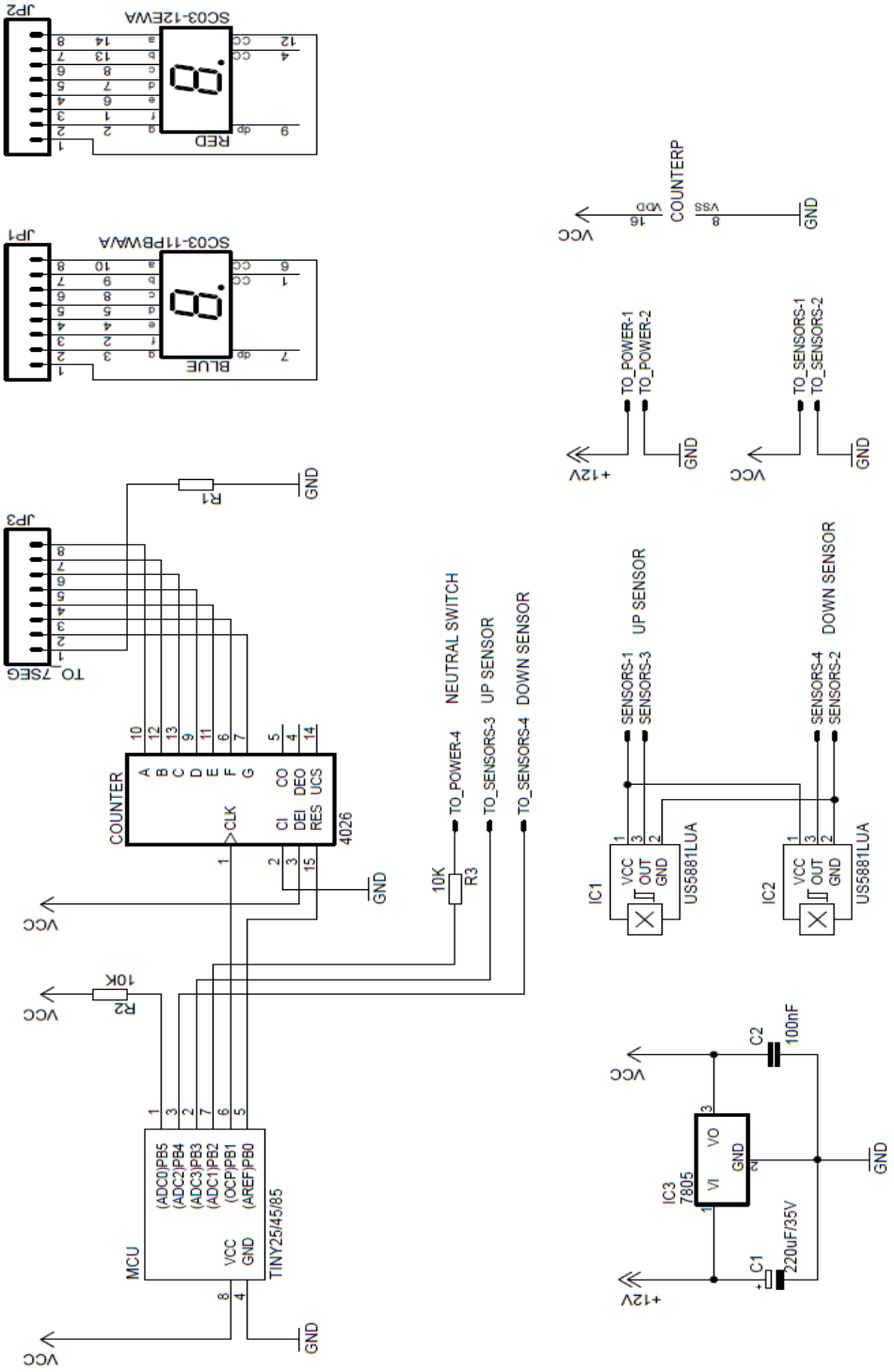


2. GLAVNI DEL

2.1 BLOK SHEMA



2.2 ELEKTRIČNA SHEMA VEZJA

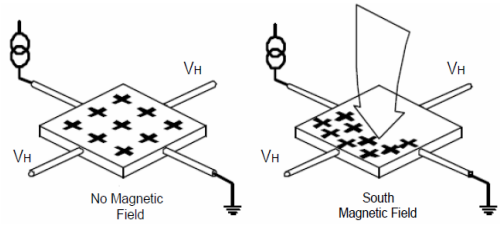


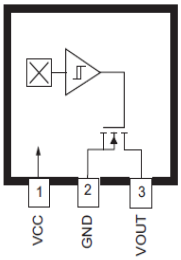
2.3 OPIS DELOVANJA

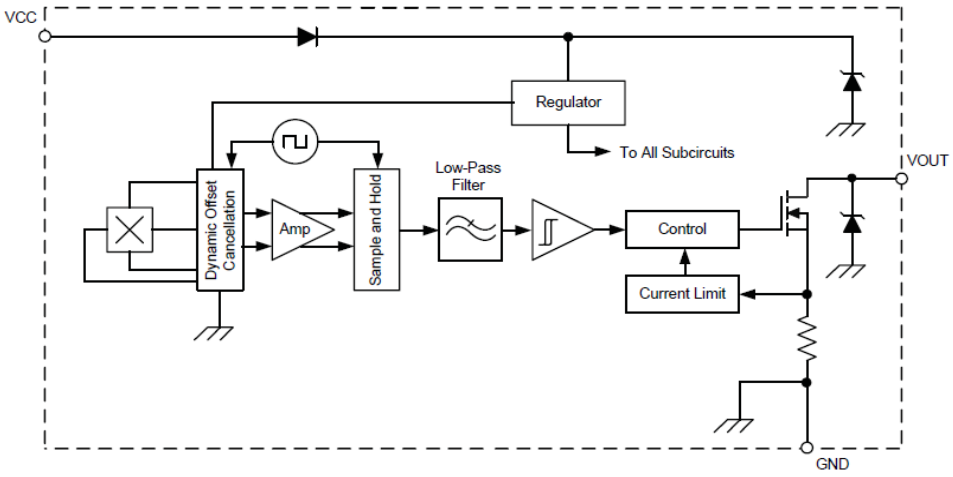
Hallova senzorja:

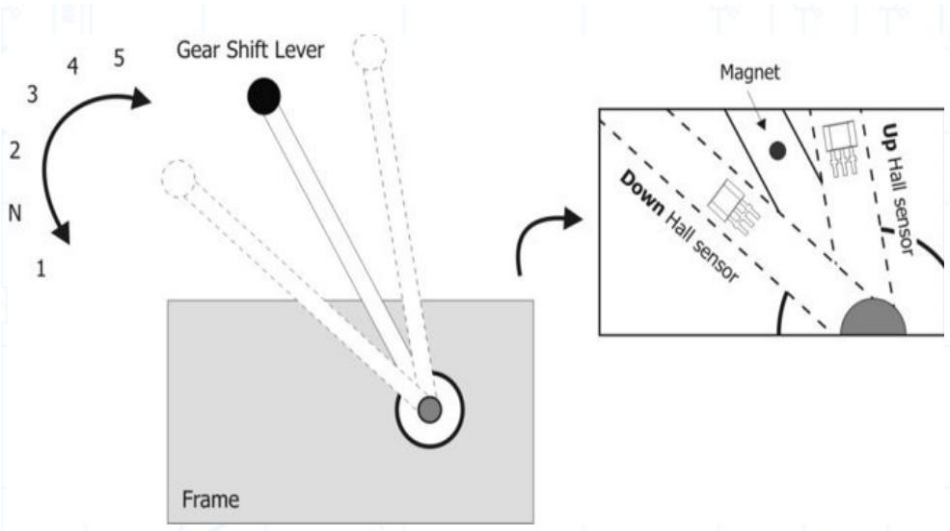
Uporabil sem unipolarni Hallov senzor A1125, kateri deluje na princip Hallovega efekta in vsebuje: napetostni regulator, Hallov napetostni regulator, malosignalni ojačevalnik, chopper stabilizacijo, Schmitov trigger in zaščito proti kratkemu stiku.

Hallova senzorja sta pritrjena na okvir motocikla tik ob ročici menjalnika.



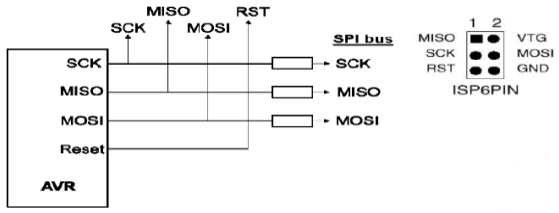


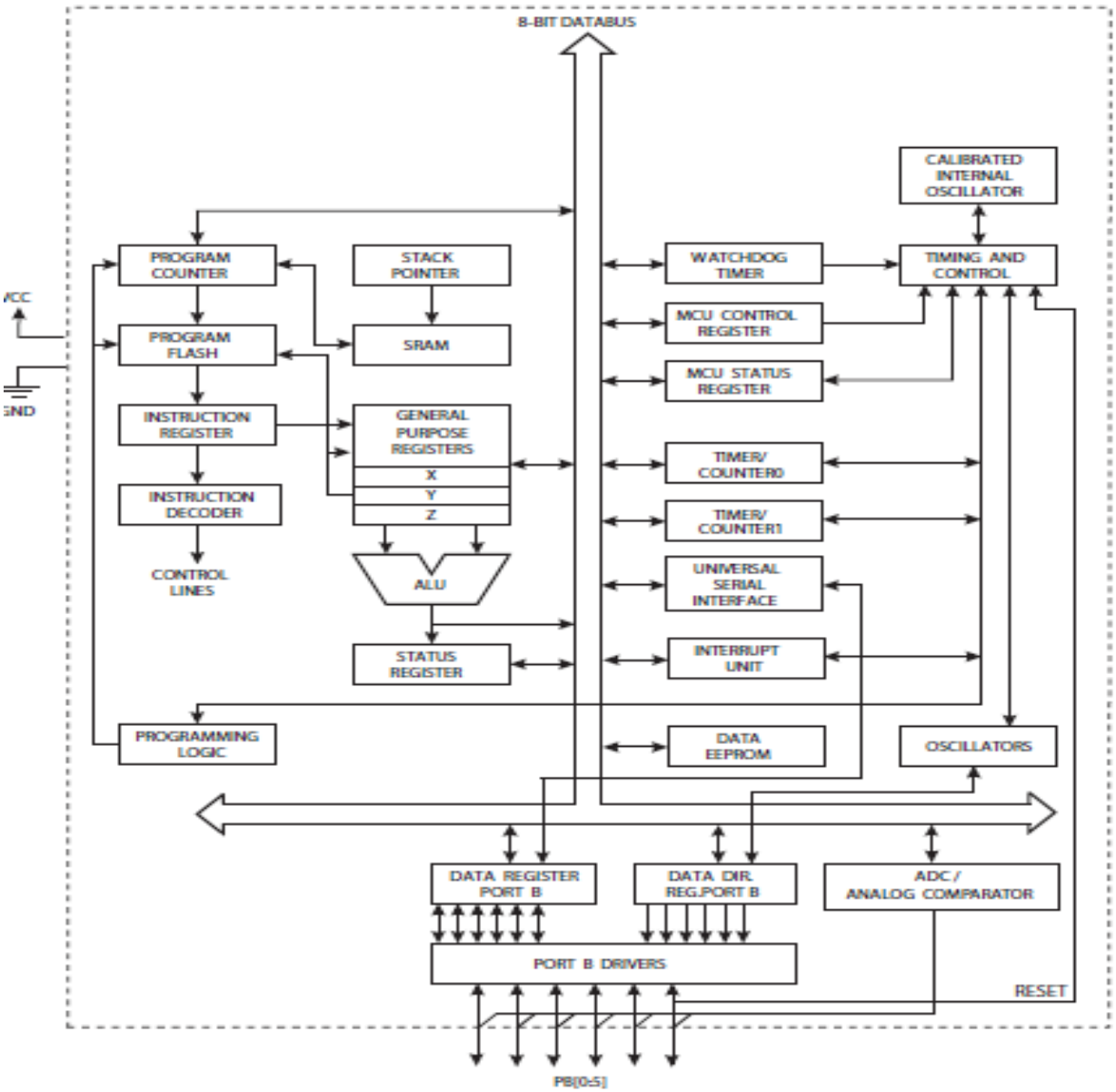




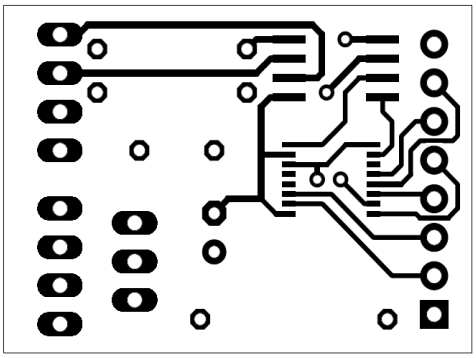
Mikrokrmilnik ATtiny25:

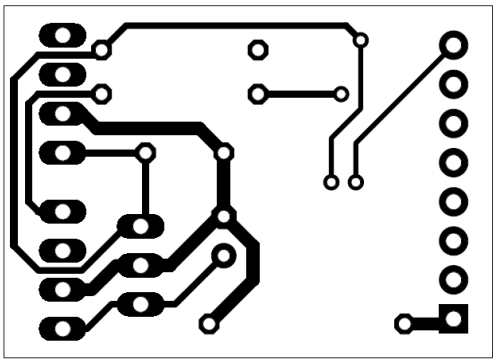
8-bitni mikrokrmilnik v CMOS tehnologiji vsebuje RISC hardversko arhitekturo in večino inštrukcij izvede v enem ciklu. Mikrokrmilnik sem sprogramiral s programatorjem AVR-Osp II, katerega sem na preizkusni ploščici priključil po spodnji sliki. Programska koda s komentarji je priložena v prilogah.

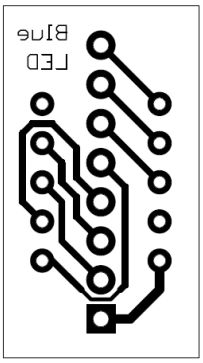


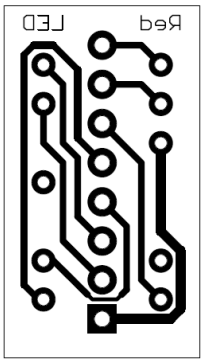


2.4 NAČRT TISKANEGA VEZJA

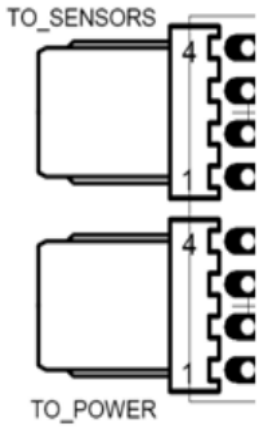








2.5 PRIKLJUČITEV SIGNALOV



TO_SENSORS_4	izhod zgornji Hallov seznor (pin3)
TO_SENSORS_3	izhod spodnji Hallov senzor(pin3)
TO_SENSORS_2	GND obeh Hallovih senzorjev (pin2)
TO_SENSORS_1	VCC(5V) obeh Hallovih senzorjev(pin1)
TO_POWER_4	nevtralno stikalo (signal iz motocikla)
TO_POWER_3	brez povezave
TO_POWER_2	baterija GND (na motociklu)
TO_POWER_1	baterija +12V (na motociklu)

2.6 KOSOVNICA

Ime	Vrednost	Element
R1	100Ω	upor
R2, R3	10KΩ, 0,5W	upor
C1	220µF, 35V	kondenzator
C2	100nF	kondenzator
MCU	ATtiny25	mikrokrmilnik
COUNTER	CD4033B	števec/dekodirnik
IC1, IC2	Hallov senzor	Hallov senzor
IC3	L7805CV	napetostni regulator
RED	Kingbright 0.3"	7-LED prikazovalnik

3. ZAKLJUČEK

Največ težav sem imel pri ročni izdelavi dvoplastne tiskanine, zato sem jo dal izdelati v bližnje podjetje, sedaj pa je vezje še na preizkusni ploščici. Vezje bi lahko izboljšal, z zamenjavo večjega 7-segmentnega prikazovalnika, zaradi boljše vidljivosti. Upor R1 bi lahko nadomestil s potenciometrom za nastavitev svetlosti prikazovalnika. Ohišje vezja bi nepredušno zaprl, da bi bilo odporno proti vodi in vlagi.

Projekt se mi je zdel zanimiv in upam da bo v praksi služil svojemu namenu. Med izdelavo sem pridobil precej praktičnega znanja in izkušenj na področju projektnega dela.

4. VIRI IN LITERATURA

[1] http://www.electronics-lab.com/projects/
[2] http://www.allegromicro.com/en/Products/Part_Numbers/1120/1120.pdf
[3] http://www.us.kingbright.com/images/catalog/SPEC/SC03-12EWA.pdf
[4] http://www.farnell.com/datasheets/3892.pdf
[5] http://focus.ti.com/lit/ds/symlink/cd4033b.pdf
[6] http://www.farnell.com/datasheets/36610.pdf
[7] http://www.magnesensor.com/pimage/081407_Hall%20IC%20Introduction.pdf

5. PRILOGE

Programska koda:

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/eeprom.h>

#define RESET_BIT PB0
#define CLOCK_BIT PB1
#define NEUTRAL_BIT PB2
#define UP_BIT PB3
#define DOWN_BIT PB4
#define RESET_DDR DDB0
#define CLOCK_DDR DDB1
#define DEBOUNCE_TIME 50*4      //Vsi časi so *4, ker ima AVR-libc 1.7.0 napako v delay funkciji
#define LOCK_TIME 250*4
#define TOP_GEAR 6
uint8_t Gear;
void ShowDigit(uint8_t Digit)
{
    uint8_t i;
    PORTB |= (1 << RESET_BIT);        // RESET pin high
    PORTB &= ~(1 << RESET_BIT);       // RESET pin low
    for (i = 0; i < Digit; i++)
    {
        PORTB |= (1 << CLOCK_BIT);    // CLOCK pin high
        PORTB &= ~(1 << CLOCK_BIT);   // CLOCK pin low
    }
}
void Init()
{
    uint8_t i;
    PORTB |= 1 << NEUTRAL_BIT;    // NEUTRAL pin enable pull-up
    PORTB |= 1 << UP_BIT;         // UP pin enable pull-up
    PORTB |= 1 << DOWN_BIT;       // DOWN pin enable pull-up
    DDRB |= 1 << RESET_DDR;       // RESET pin as output
    DDRB |= 1 << CLOCK_DDR;       // CLOCK pin as output
    PORTB &= ~(1 << RESET_BIT);   // RESET pin low
    // Self test effect
    ShowDigit(0);
    _delay_ms(500*4);
    for (i = 1; i < 10; i++)
    {
        ShowDigit(i);
        _delay_ms(100*4);
    }
```

```
45      ShowDigit(0);
46      _delay_ms(500*4);
47      Gear = eeprom_read_byte((uint8_t*)0);  // Read initial value from EEPROM
48      ShowDigit(Gear);                       // Show initial value
49  }
50  int Released(uint8_t Bit)
51  {
52      if (bit_is_clear(PINB,Bit))
53      {
54          _delay_ms(DEBOUNCE_TIME);
55          if (bit_is_set(PINB,Bit)) return 1;
56      }
57
58      return 0;
59  }
60  int main(void)
61  {
62      Init();
63      while(1)
64      {
65          if (Released(UP_BIT))
66          {
67              if ((Gear > 0) && (Gear < TOP_GEAR)) Gear++;
68              if (Gear <= 0) Gear = 2;
69              if (Gear > 6) Gear =6 ;
70              if (bit_is_clear(PINB,NEUTRAL_BIT)) Gear = 0;
71              eeprom_write_byte((uint8_t*)0,Gear);
72              ShowDigit(Gear);
73              //_delay_ms(1000*4);
74              _delay_ms(LOCK_TIME);
75          }
76          if (Released(DOWN_BIT))
77          {
78              if (Gear > 1) Gear--;
79              if (Gear <= 0) Gear = 1;
80
81              if (bit_is_clear(PINB,NEUTRAL_BIT)) Gear = 0;
82
83              eeprom_write_byte((uint8_t*)0,Gear);
84              ShowDigit(Gear);
85              _delay_ms(LOCK_TIME);
86          }
87      }
88  }
```