1. Project title:

Seven segment display self countdown and motor rotation detection using Hall sensor

Pântea Bogdan (bogdanp_2008@yahoo.com)

Fuiorea-Novac Constantin (fuiorea.constantin93@gmail.com)

2. Abstract

Motor rotation detection using Hall sensor and seven segment display counting.

3. Introduction, project aims and objectives

This project's objective is to detect each rotation of a DC motor using a Hall sensor and with a seven segment display the user can show a one digit number. Next to the seven display segment are four leds on which we can read the one digit number in binary.

4. System overview

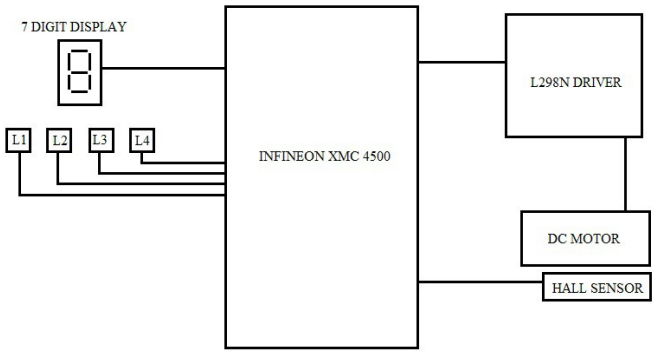
XMC4500 can control the Hall sensor, the seven segment display, the four leds and the L298N motor driver which controls the DC motor.

The motor driver can control the DC motor via two digital pins and next to the motor there is strategically placed the Hall sensor close enough such that it can detect the magnet which is soldered on the motor's wheel.

The seven segment can display a number between 0 and 9, but if the number is higher than 9 the letter E will be displayed meaning an error and the four leds will be all switched on.

5. Schematics and components

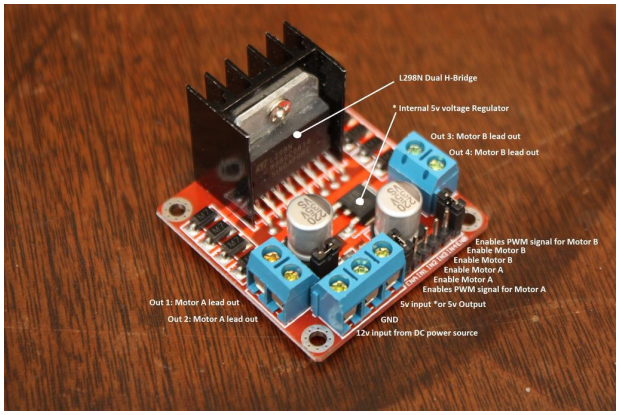
Main schematic



Infineon XMC4500 Relax

Infineon XMC4500 Relax Evaluation Kit is equipped with the ARM® Cortex™-M4 based XMC4500 microcontroller (MCU) from Infineon Technologies. This kit is designed to evaluate the capabilities of the XMC4500 MCU and the powerful, free-of-charge tool chain DAVE™3. The XMC4500 Relax Kit features an Ethernet-enabled communication option, e.g. to run an embedded web server. The user can store their own HTML web pages on a microSD Card or control the XMC4500 via the web browser on their PC. .

L298N - Motor Driver

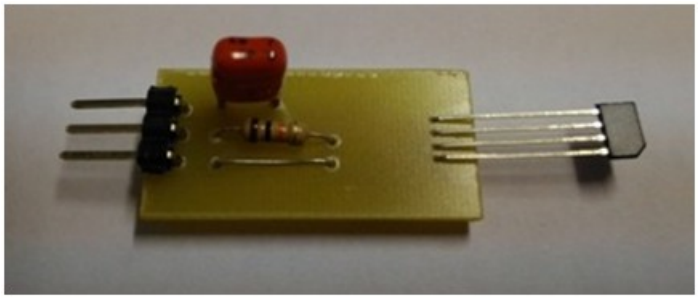


This is the hardware device which can run ther DC motor.

DC Motor

A DC motor is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor. Most types produce rotary motion; a linear motor directly produces force and motion in a straight line

Hall Sensor



A Hall effect sensor is a transducer that varies its output voltage in response to a magnetic field. Hall effect sensors are used for proximity switching, positioning, speed detection, and current sensing applications.

In its simplest form, the sensor operates as an analog transducer, directly returning a voltage. With a known magnetic field, its distance from the Hall plate can be determined. Using groups of sensors, the relative position of the magnet can be deduced

6. Software

```
#include <DAVE.h>
/**
* A,B,C,D,E,F,G pins used for 7 segment digit
* Buton1, Buton2 pins for the buttons on XMC4500
* LED1,LED2,LED3,LED4 pins for binary counter
* MIN1,MIN2 pins used for DC motor
* VAL_SENZOR pin that return the value when the magnet is detected by the Hall sensor
* LED pins used to signal when the magnet is detected
*/
void digit(int numar)
{
switch(numar)
{
case 0:
DIGITAL_IO_SetOutputLow(&G);
DIGITAL_IO_SetOutputHigh (&F);
DIGITAL_IO_SetOutputHigh (&E);
DIGITAL_IO_SetOutputHigh (&D);
DIGITAL_IO_SetOutputHigh (&C);
DIGITAL_IO_SetOutputHigh (&B);
DIGITAL_IO_SetOutputHigh (&A);
DIGITAL_IO_SetOutputLow(&LED1);
DIGITAL_IO_SetOutputLow(&LED2);
DIGITAL_IO_SetOutputLow(&LED3);
DIGITAL_IO_SetOutputLow(&LED4);
break;
case 1:
DIGITAL_IO_SetOutputLow(&G);
```

```
DIGITAL_IO_SetOutputLow(&F);
DIGITAL_IO_SetOutputLow(&E);
DIGITAL_IO_SetOutputLow(&D);
DIGITAL_IO_SetOutputHigh (&C);
DIGITAL_IO_SetOutputHigh (&B);
DIGITAL_IO_SetOutputLow(&A);
DIGITAL_IO_SetOutputLow(&LED1);
DIGITAL_IO_SetOutputLow(&LED2);
DIGITAL_IO_SetOutputLow(&LED3);
DIGITAL_IO_SetOutputHigh(&LED4);
break;
case 2:
DIGITAL_IO_SetOutputHigh (&G);
DIGITAL_IO_SetOutputLow(&F);
DIGITAL_IO_SetOutputHigh (&E);
DIGITAL_IO_SetOutputHigh (&D);
DIGITAL_IO_SetOutputLow(&C);
DIGITAL_IO_SetOutputHigh (&B);
DIGITAL_IO_SetOutputHigh (&A);
DIGITAL_IO_SetOutputLow(&LED1);
DIGITAL_IO_SetOutputLow(&LED2);
DIGITAL_IO_SetOutputHigh(&LED3);
DIGITAL_IO_SetOutputLow(&LED4);
break;
case 3:
DIGITAL_IO_SetOutputHigh (&G);
DIGITAL_IO_SetOutputLow(&F);
DIGITAL_IO_SetOutputLow(&E);
DIGITAL_IO_SetOutputHigh (&D);
DIGITAL_IO_SetOutputHigh (&C);
DIGITAL_IO_SetOutputHigh (&B);
DIGITAL_IO_SetOutputHigh (&A);
DIGITAL_IO_SetOutputLow(&LED1);
DIGITAL_IO_SetOutputLow(&LED2);
DIGITAL_IO_SetOutputHigh(&LED3);
DIGITAL_IO_SetOutputHigh(&LED4);
break;
case 4:
DIGITAL_IO_SetOutputHigh (&G);
DIGITAL_IO_SetOutputHigh (&F);
DIGITAL_IO_SetOutputLow(&E);
DIGITAL_IO_SetOutputLow(&D);
DIGITAL_IO_SetOutputHigh (&C);
DIGITAL_IO_SetOutputHigh (&B);
DIGITAL_IO_SetOutputLow(&A);
DIGITAL_IO_SetOutputLow(&LED1);
DIGITAL_IO_SetOutputHigh(&LED2);
DIGITAL_IO_SetOutputLow(&LED3);
DIGITAL_IO_SetOutputLow(&LED4);
break;
case 5:
DIGITAL_IO_SetOutputHigh (&G);
DIGITAL_IO_SetOutputHigh (&F);
DIGITAL_IO_SetOutputLow(&E);
```

```
DIGITAL_IO_SetOutputHigh (&D);
DIGITAL_IO_SetOutputHigh (&C);
DIGITAL_IO_SetOutputLow(&B);
DIGITAL_IO_SetOutputHigh (&A);
DIGITAL_IO_SetOutputLow(&LED1);
DIGITAL_IO_SetOutputHigh(&LED2);
DIGITAL_IO_SetOutputLow(&LED3);
DIGITAL_IO_SetOutputHigh(&LED4);
break;
case 6:
DIGITAL_IO_SetOutputHigh (&G);
DIGITAL_IO_SetOutputHigh (&F);
DIGITAL_IO_SetOutputHigh (&E);
DIGITAL_IO_SetOutputHigh (&D);
DIGITAL_IO_SetOutputHigh (&C);
DIGITAL_IO_SetOutputLow(&B);
DIGITAL_IO_SetOutputHigh (&A);
DIGITAL_IO_SetOutputLow(&LED1);
DIGITAL_IO_SetOutputHigh(&LED2);
DIGITAL_IO_SetOutputHigh(&LED3);
DIGITAL_IO_SetOutputLow(&LED4);
break;
case 7:
DIGITAL_IO_SetOutputLow(&G);
DIGITAL_IO_SetOutputLow(&F);
DIGITAL_IO_SetOutputLow(&E);
DIGITAL_IO_SetOutputLow(&D);
DIGITAL_IO_SetOutputHigh (&C);
DIGITAL_IO_SetOutputHigh (&B);
DIGITAL_IO_SetOutputHigh (&A);
DIGITAL_IO_SetOutputLow(&LED1);
DIGITAL_IO_SetOutputHigh(&LED2);
DIGITAL_IO_SetOutputHigh(&LED3);
DIGITAL_IO_SetOutputHigh(&LED4);
break;
case 8:
DIGITAL_IO_SetOutputHigh (&G);
DIGITAL_IO_SetOutputHigh (&F);
DIGITAL_IO_SetOutputHigh (&E);
DIGITAL_IO_SetOutputHigh (&D);
DIGITAL_IO_SetOutputHigh (&C);
DIGITAL_IO_SetOutputHigh (&B);
DIGITAL_IO_SetOutputHigh (&A);
DIGITAL_IO_SetOutputHigh(&LED1);
DIGITAL_IO_SetOutputLow(&LED2);
DIGITAL_IO_SetOutputLow(&LED3);
DIGITAL_IO_SetOutputLow(&LED4);
break;
case 9:
DIGITAL_IO_SetOutputHigh (&G);
DIGITAL_IO_SetOutputHigh (&F);
DIGITAL_IO_SetOutputLow(&E);
DIGITAL_IO_SetOutputHigh (&D);
DIGITAL_IO_SetOutputHigh (&C);
```

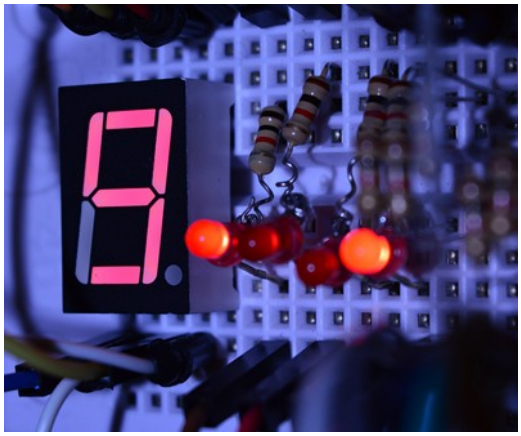
```
DIGITAL_IO_SetOutputHigh (&B);
DIGITAL_IO_SetOutputHigh (&A);
DIGITAL_IO_SetOutputHigh(&LED1);
DIGITAL_IO_SetOutputLow(&LED2);
DIGITAL_IO_SetOutputLow(&LED3);
DIGITAL_IO_SetOutputHigh(&LED4);
break;
case 10:
DIGITAL_IO_SetOutputHigh (&G);
DIGITAL_IO_SetOutputHigh (&F);
DIGITAL_IO_SetOutputHigh (&E);
DIGITAL_IO_SetOutputHigh (&D);
DIGITAL_IO_SetOutputLow (&C);
DIGITAL_IO_SetOutputLow (&B);
DIGITAL_IO_SetOutputHigh (&A);
DIGITAL_IO_SetOutputHigh(&LED1);
DIGITAL_IO_SetOutputHigh(&LED2);
DIGITAL_IO_SetOutputHigh(&LED3);
DIGITAL_IO_SetOutputHigh(&LED4);
break;
}
}

void delay() // delay fabricat
{
for(long j=0;j<4000000;j++);
}

void delayForDecrease() // delay fabricat
{
for(long j=0;j<10000000;j++);
}

int counter=0;

void first_Thread()
{
while(counter!= -1)
{
delayForDecrease();
digit(counter);
counter--;
}
}

void second_Thread()
{
DIGITAL_IO_SetOutputHigh (&MIN1);
DIGITAL_IO_SetOutputLow (&MIN2);
if(!DIGITAL_IO_GetInput (&VAL_SENZOR))
DIGITAL_IO_SetOutputHigh (&LED);
else
```

```
{
DIGITAL_IO_SetOutputLow (&LED);
}
}

int main(void)
{
 DAVE_STATUS_t status;
 status = DAVE_Init();           /* Initialization of DAVE APPs  */

 if(status == DAVE_STATUS_FAILURE)
 {
   /* Placeholder for error handler code. The while loop below can be replaced with an user error handler. */
   XMC_DEBUG("DAVE APPs initialization failed\n");

   while(1U)
   {

   }
 }
 /* Placeholder for user application code. The while loop below can be replaced with user application code. */
 while(1U)
 {
 while(1)
 {
 if(!DIGITAL_IO_GetInput (&Buton1)) // buton  apasat
 {
 if(counter<=9)
 {
 counter++;
 digit(counter);
 delay();
 }
 else digit(10);
 }
 if(!DIGITAL_IO_GetInput (&Buton2)) break;
 }
 first_Thread();
 while(!DIGITAL_IO_GetInput (&Buton2) &&!DIGITAL_IO_GetInput (&Buton1))
 {
```

```
 second_Thread();
 }
 DIGITAL_IO_SetOutputHigh (&MIN1);
 DIGITAL_IO_SetOutputHigh (&MIN2);

 }
}
```

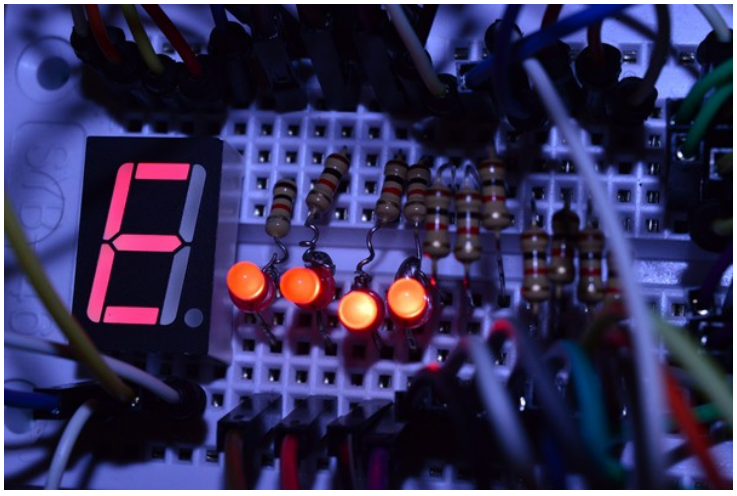
7. Project results & applications

Using the XMC4500 button 1 the user can increment up to number 9 and it will be displayed on the seven segment display; the four leds next to it will display the number in binary.

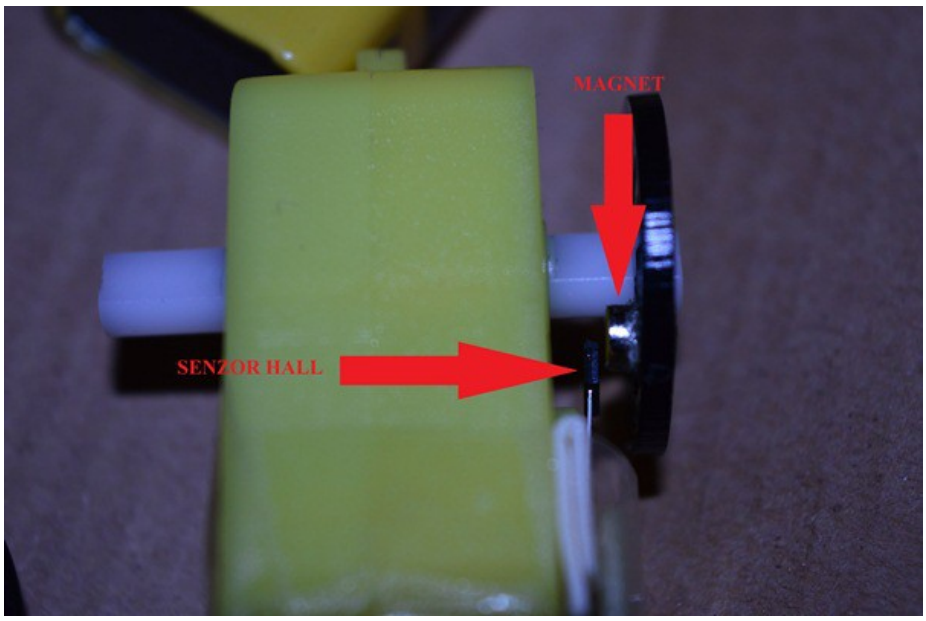


By pressing the XMC4500 button 2 the countdown will begin from the number displayed on the seven segment.

In case of a number higher than 9 is incremented, the seven segment display will show a letter E (error) and the four leds next to it will be all switched on warning the user.



When the countdown is finished, by pressing simultaneously the buttons 1 and 2 the motor will start spinning a wheel on which is attached a small but powerfull magnet that will passing next to the Hall sensor. Each time the sensor will “feel” the magnet, the red led on the XMC4500 will blink.



8. Reference

http://embedac.ro
www.wikipedia.com