1. Project title: Stepper motor control using a Hall sensor with the XMC1100 Boot kit development board

Mihalceanu Radu-Daniel
Email:mihalceanu_daniel@yahoo.com

2. Abstract

Using the XMC 1100 Boot kit I developed an application to control a stepper motor, in accordance with the digital value, that was read from a digital input. The stepper motor was rotating clockwise or counterclockwise, depending on the presence of a magnet near the Hall sensor or away from the sensor.

3. Introduction, project aims and objectives

The XMC1100 CPU Card for ArduinoTM has two rows of pin headers which fully compatible with ArduinoTM shield. Hence, user can buy various Arduino shield boards off-the-shelf to test the capabilities of XMC1100 Microcontroller.
The XMC1100 CPU Card for ArduinoTM is equipped with the following features:

- XMC1100 (ARM® CortexTM -M0 based) Microcontroller, TSSOP38
- Headers compatible with ArduinoTM shield
- Detachable SEGGER J-Link debugger and UART virtual COM port, with micro USB connector
- Power supply concept compatible with ArduinoTM Uno
- One LED as required from ArduinoTM board specification and six additonal LEDs.

The project aim is to control a stepper motor, in a clockwise direction or a counterclockwise direction, depending on the digital value of a Hall sensor, which is read from an analog-digital input pin from the development board.
The applications that can be made using a Hall sensor, nowadays, are for: proximity(distance) detection, power sensing, speed detection, and current sensing. Application examples are for timing

the speed of wheels and shafts, alarm systems, keyboard and printers. This kind of application can be used to count the speed and the number of rotations per minute of a car wheel and also to change the direction or speed of a stepper motor, depending on the nearby magnetic field, detected with a Hall sensor.
The unipolar stepper motor system provides accurate positioning with the lowest parts count and by lowest parts count and bill of materials cost.
Computer controlled stepper motors are a type of motion-control positioning system. They are typically digitally controlled as part of an open loop system for use in holding or positioning applications. In the field of lasers and optics they are frequently used in precision positioning equipment such as linear actuators, rotation stages, goniometers and mirror mounts. Other uses are in packaging machinery, and positioning of valve pilot stages for fluid control systems. For commercial purposes, stepper motors are used in floppy disk drives, flatbed scanners, computer printers, plotters, image scanners, compact disk drives, intelligent lighting, camera lenses and also, more recently, in 3D printers.

4. System overview

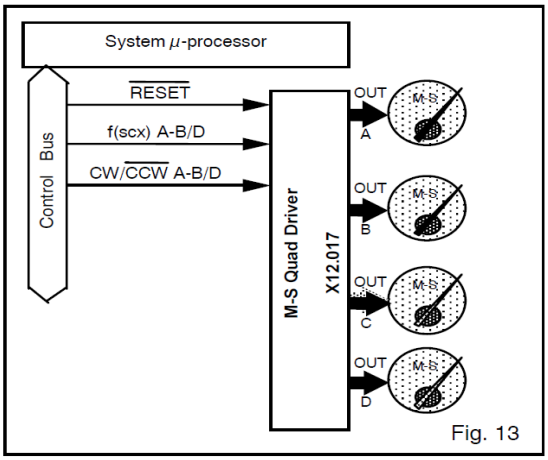
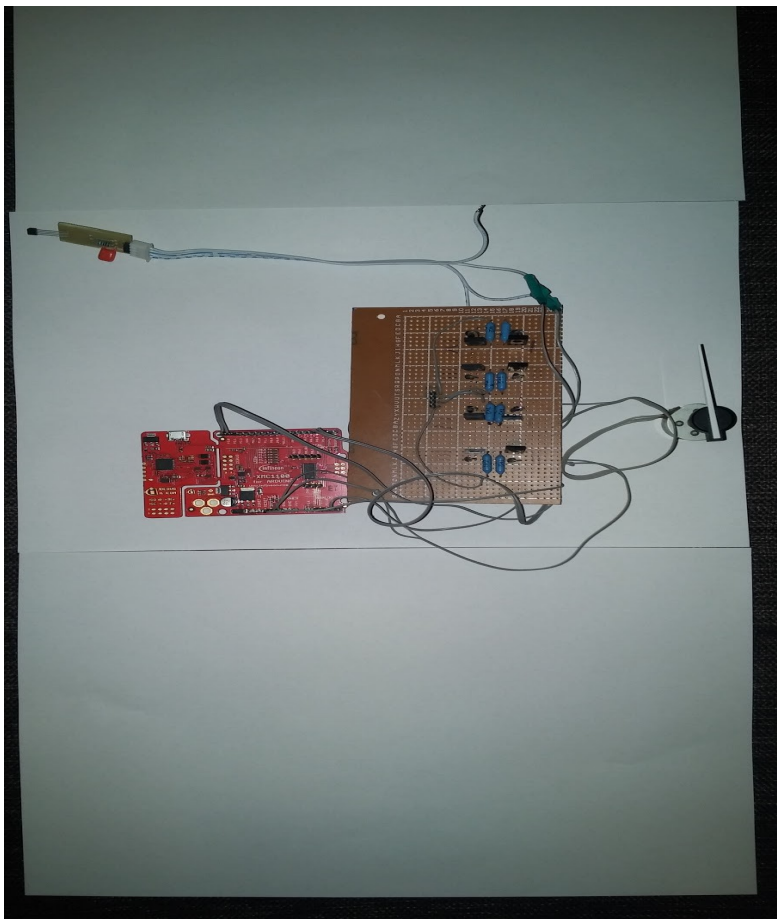


Fig. 13

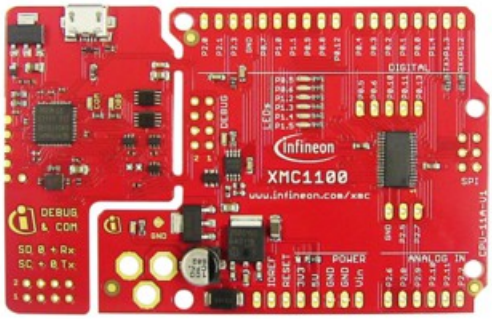
The system consists of the XMC1100 board, to which is connected a buffer, for controlling the stepper motor. To the board, at the analog-digital input pin P2.2 there is connected the Hall sensor. The 2 coils of the motor are connected in the following manner: one coil is connected with wires to the P2.0 and P2.1 pins and the other coil is connected to the P0.0 and P1.4 pins. The system functions as follows: First the microcontroller, reads the digital value, from the Hall sensor, and if there is no magnetic field near the sensor, it will give the command to the motor to rotate at the right. If in this time we approach a magnet near the Hall sensor, then the microcontroller changes the motor's direction to the left and keeps this direction until we move away the magnet and then we bring it near the sensor again. If this happens, then the microcontroller gives the command to the step motor to change it's direction to the right, in a clockwise direction.
The overview of the system is shown in the following images:

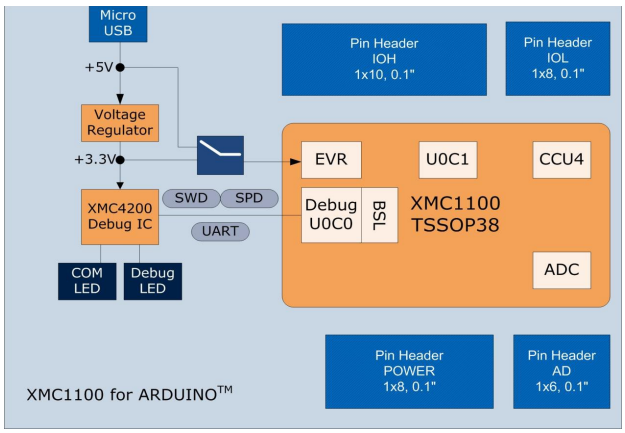


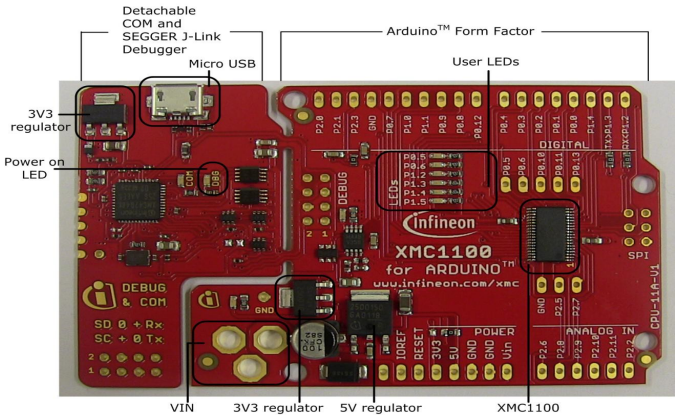
5. Schematics and components

For the implementation of the application, I used the following components:

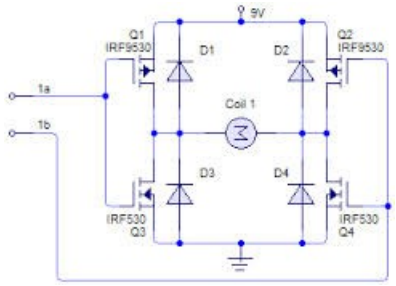
- the XMC1100 Boot Kit development board



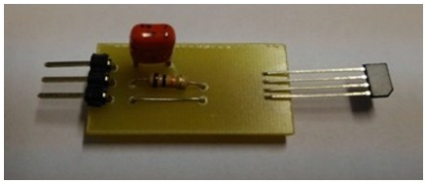




- the H-Bridge buffer for controlling the stepper motor:



- 4 by 4 NPN and PNP transistors;
- resistors;
- 4 diodes;
- a TLE4921-5U Hall sensor:



- a stepper motor taken from a floppy-disk drive

6. Software

The IDE that I used is DAVE version 4.1.4. **DAVE (Infineon)** Digital Application Virtual Engineer

(DAVE™), a C/C++-language software development and code generation tool for microcontroller applications. DAvE is a standalone system with automatic code generation modules and is suited to develop software drivers for Infineon microcontrollers and aids the developer with automatically created C-level templates and user desired functionalities.

Latest releases of DAVE include all required parts to develop code, compile and debug on the target for free (based on the ARM gcc toolsuite). Together with several low-cost development boards one can get involved in microcontroller design very easy. This makes Infineon microcontroller products also more usable to small companies and to home-use / DIY projects - similar to established products of Atmel (AVR, SAM) and Microchip (PIC, PIC32) to name a few.

The successor of the Eclipse-based development environment for C/C++ and/or GUI-based development using "Apps". It generates code for the latest XMC1xxx and XMC4xxx microcontrollers using Cortex-M processors. The code generation part is significantly improved. Beside the free DAVE™ development software, a DAVE™ SDK is a free development environment to set up own "Apps" for DAVE™. Details (downloads, getting started, tutorials etc.) can be found on the website.

After starting DAVE, an Eclipse environment appears. In the project browser, a standard C/C++ or a DAVE project can be set up by selecting one of the available processors of Infineon. The latter project setup allows configuration of the selected MCU using a GUI-based approach. This simplifies the setup of complex peripherals significantly. It also ensures that related peripherals (e.g. PLL configuration for peripheral clock on the one hand and the peripheral itself on the other hand) are configured consistently. In simplest cases, the user ends up in coding callback functions to certain peripheral events and a main loop. Pre-configured project templates allows to successfully set up first programs easily.

The code that I used to control the stepper motor, depending on the digital value read from the Hall sensor is shown next:

```
/*
Conexiuni: bobina A:  P1.4, P0.0
           bobina B: P2.1, P2.0
*/
```

```
#include <DAVE.h>                 //Declarations from DAVE Code
Generation (includes SFR declaration)
/**
* @brief main() - Application entry point
*
* <b>Details of function</b><br>
* This routine is the application entry point. It is invoked by
the device startup code. It is responsible for
* invoking the APP initialization dispatcher routine - DAVE_Init()
and hosting the place-holder for user application
* code.
*/
int j=6;
int k,l,t;
void delay(long unsigned int i) // delay fabricat
{ while(i--)
{
j++;
j--;
}
}
int main(void)
{
 DAVE_STATUS_t status;
 uint32_t pin_rotire;
 bool out=0;
 uint32_t x=0;
 bool val_veche=0;
 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. */
 DIGITAL_IO_SetOutputHigh (&a1); DIGITAL_IO_SetOutputHigh (&a2);
 DIGITAL_IO_SetOutputHigh (&b1); DIGITAL_IO_SetOutputHigh (&b2);
 t=8000;
 pin_rotire=DIGITAL_IO_GetInput(&sens_rotatie);
while(1U)
{
pin_rotire=DIGITAL_IO_GetInput(&sens_rotatie);
delay(3000);
if(pin_rotire==1)
{
x++;
}
else
{
if(x%2==0)
{
DIGITAL_IO_SetOutputLow (&a2); //
delay(t);
DIGITAL_IO_SetOutputHigh (&a2); // curent i1
DIGITAL_IO_SetOutputLow (&led_clipitor_dreapta); //
delay(t);
DIGITAL_IO_SetOutputHigh (&led_clipitor_dreapta);
DIGITAL_IO_SetOutputLow (&b2); //
delay(t);
DIGITAL_IO_SetOutputHigh (&b2); // curent i3
DIGITAL_IO_SetOutputLow (&a1);
delay(t);
DIGITAL_IO_SetOutputHigh (&a1); // curent i2
DIGITAL_IO_SetOutputLow (&b1); delay(t);
DIGITAL_IO_SetOutputHigh (&b1); // curent i4
} // rotatie drepta
else
{

DIGITAL_IO_SetOutputLow (&b1);
```

```
delay(t);
DIGITAL_IO_SetOutputHigh (&b1); // curent i4
DIGITAL_IO_SetOutputLow (&led_clipitor_stanga); //
delay(t);
DIGITAL_IO_SetOutputHigh (&led_clipitor_stanga);
DIGITAL_IO_SetOutputLow (&a1); //
delay(t);
DIGITAL_IO_SetOutputHigh (&a1); // curent i2
DIGITAL_IO_SetOutputLow (&b2); //
delay(t);
DIGITAL_IO_SetOutputHigh (&b2); // curent i3
DIGITAL_IO_SetOutputLow (&a2); //
delay(t);
DIGITAL_IO_SetOutputHigh (&a2); // curent i1
} // rotatie stanga
}
}
}
```

7. Project results & applications

The application works approximately the way it should, in that it changes the direction of the motor when a magnet is approached to the Hall sensor and maintains the direction, until the magnet is brought close to the sensor again.

8. Reference

http://embedac.ro/Infineon/Lab/L4/Laborator4.htm

http://embedac.ro/Infineon/Lab/L4/InfineonBoard_Users_Manual_XMC1100_CPU_Card_R2.PDF

https://code.google.com/p/arduino-to-xmc/wiki/UnipolarStepperMotor

https://en.wikipedia.org/wiki/DAvE_(Infineon)

https://itp.nyu.edu/physcomp/labs/motors-and-transistors/lab-controlling-a-stepper-motor-with-an-h-bridge/

http://homepage.cs.uiowa.edu/~jones/step/circuits.html

http://www.infineonforums.com/threads/1669-XMC1100-Boot-Kit-Getting-Started-and-Examples-Arduino-to-XMC