Diploma Thesis

Andreas Gruber

September 30, 2012

Abstract

The amount of robots in our life rises very fast. They build our cars, they mow the lawn, they do the hoovering and a lot of other things which most people do not even recognize. Most of these intelligent systems which are mobile systems, are connected to a central control station or are remotely controlled. In some cases, however, it is impossible to connect the mobile system to a control device or the mobile system needs to act very fast and, therefore, needs to calculate its behaviour by its own. In this case a device needs to know its current position, speed, and direction. Also a lot of other properties need to be known to compute a behaviour which is as good as possible for the current situation. To find out the values of these properties the robot needs a lot of sensors. In order to keep the production costs low, the mass of these sensors is pretty cheap which, in turn, rises the problem that those can produce errors and they even can brake while the device is working. In one of these cases the device should detect that the sensor doesn't work any more and replace its results by the result of other sensors.

The main goal of this thesis is to build a system which calculates the most accurate current position of a model car based on a set of different (cheap) sensors. To take more cheap sensors has a lot of pros. First the whole device can be produced very cheaply. Another very important consequence is that other cheap sensors help to improve the results. The most important advantage (??) is that errors and measuring faults can be detected and the wrong values can be replaced by correct ones. Systems like this have a big application spectrum in future because autonomous robots will be needed in more and more branches.

Write about three sentences about the co-operation with FLLL in Hagenberg.

- There is a co-operation, who is the contact person
- What is the FLLL working in?
- What is the general project you are working in? How does your thesis fit into the general work of the FLLL?

Contents

Chapter 1

Environment

1.1 AVR

AVR is the name of a microcontroller-family which is produces by the company Atmel. Atmel produces different types of microcontroller. The different types can be grouped in:

32-bit AVR UC3	
AVR XMEGA	
Automotive AVR	
megaAVR	
tinyAVR	(Small, cheap microcontroller with a power supply down to 0,7 V)
Battery Management	(Battery management chips for Li-ion batteries)

During this diploma thesis I mostly using megaAVR or ATmega.

1.1.1 ATmega 2560

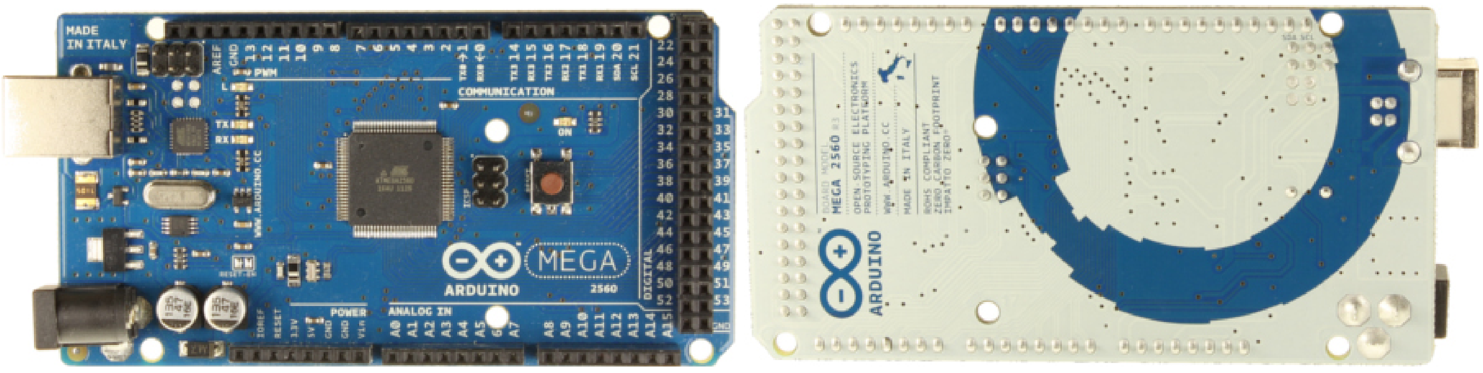
ATmega 2560 is part of many different electronic projects. It is a very powerful microcontroller wich has a lot of different hardware implemented features.

Summary

Speed Grade	0-16 MHz
Flash Memory	256 KB
SRAM	8 KB
EEPROM	4 KB
I/O Lines	86
8-bit Timer	2
16-bit Timer	4
4-bit PWM	4
2 to 16-bit PWM	12
10-bit ADC	16
USART / UART	4
TWI / I2C	1

1.2 Arduino

1.2.1 Arduino Mega 2560



The Arduino Mega 2560 is a microcontroller board based on the ATmega 2560. It contains the full hardware to programm the microcontroller. So a USB to serial converter is installed on the board. The board also contains also a voltage regulator to supply it by an external power supply
The design has the advantage that the shields from the Arduino UNO are compatible to the Arduino Mega

1.2.2 Datatypes

data type	Bit	value			
		min	max	min	max
int8_t, signed char	8	-128	127	-2^7	$2^7 - 1$
int16_t, signed short, signed int	16	-32768	32767	-2^{15}	$2^{15} - 1$
int32_t, signed long	32	-2147483648	2147483647	-2^{31}	$2^{31} - 1$
int64_t, signed long long	64	-9223372036854775808	9223372036854775807	-2^{63}	$2^{63} - 1$
uint8_t, unsigned char	8	0	255	0	$2^8 - 1$
uint16_t, unsigned short, unsigned int	16	0	65535	0	$2^{16} - 1$
uint32_t, unsigned long	32	0	4294967295	0	$2^{32} - 1$
uint64_t, unsigned long long	64	0	18446744073709551615	0	$2^{64} - 1$
float, double	32	$-3.4028235 * 10^{38}$	$3.4028235 * 10^{38}$		

1.2.3 Microcontroller I/O

Digital input Pins

```
value = digitalRead(pin)
```

Digital output Pins

```
digitalWrite(pin, value)
```

Analog inupt Pins

```
value = analogRead(Apin)
```

Value has a resolution of 10 Bit. So analog read will return a value between 0 and 1024.

Analog Output / PWM Pins

```
analogWrite(PWMpin, value)
```

Value have to be between 0 (off) and 255 (on)

UART

Before the serial connection is used it have to be started:

```
Serial.beginn(speed);
```

Then bytes can be written to ther serial port:

```
Serial.write(value);
```

Or something can be printed to the serial port:

```
Serial.println(value);
```

A small code snippet for a serial connection could look like this:

```
Serial.beginn(9600);
Serial.println("Hello World");
byte in;
while(Serial.available() > 0)
  in = Serial.read();
char c = Serial.read();
```

I2C

At first the I2C bus have to be started:
`Wire.beginn(address);`
Then a request can be started:
`Wire.requestFrom(device,size);`
At least bytes can be read from the device:
`byte b = Wire.read();`
Another option is to start a transmission: `Wire.beginTransmission(device);`
Then it is possible to write data to the communication partner:
`Wire.write(value);`
After sending the data the transmmission have to be closed:
`Wire.endTransmission();`
A example for a I2C connection could look like this:

```
Wire.begin()
Wire.requestFrom(0x23,1);
byte b = Wire.read();
Wire.beginTransmission(0x23);
Wire.write(b);
Wire.endTransmission();
```
