mobile tracking car

System Architecture & Design, HW

Multidisciplinary Project for the Applied App Development courses (ITIPRJ)

Supervisor: MSc. Torben Gregersen

Group: AAD1

Editor: Berger Andreas, Pfister Michael, François Grzybowski, Constantin Nivet, Noel Peña, Jorge Mañanas Fernández

December 9, 2014

Document History

Version	Date	Initials	Description
1.0	25.11.2014	**BA**	document created
1.1	26.11.2014	**BA**	added descriptions
1.2	27.11.2014	**BA**	added descriptions
1.3	28.11.2014	**BA**	added new section
1.4	30.11.2014	**BA**	finalized the document for skimming
1.5	09.12.2014	**BA** & **PM**	some improvements & finalizing for hand-in

Table 1: Document History

Approval

Author(s):	Berger Andreas - [**BA**]
	Pfister Michael - [**PM**]
	Jorge Mañanas Fernández - [**JM**]
	Constantin Nivet - [**CN**]
	Noel Peña - [**NP**]
	François Grzybowski - [**FG**]
Project Name:	mobile tracking car
Document ID:	system-architecture-design-hw.pdf

Table 2: Approval

Contents

1 Information

This document is based on the requirement-specification.pdf document.

2 General HW Block Diagram for the System

The general purpose for this document is to describe the different interfaces which are used in our project.

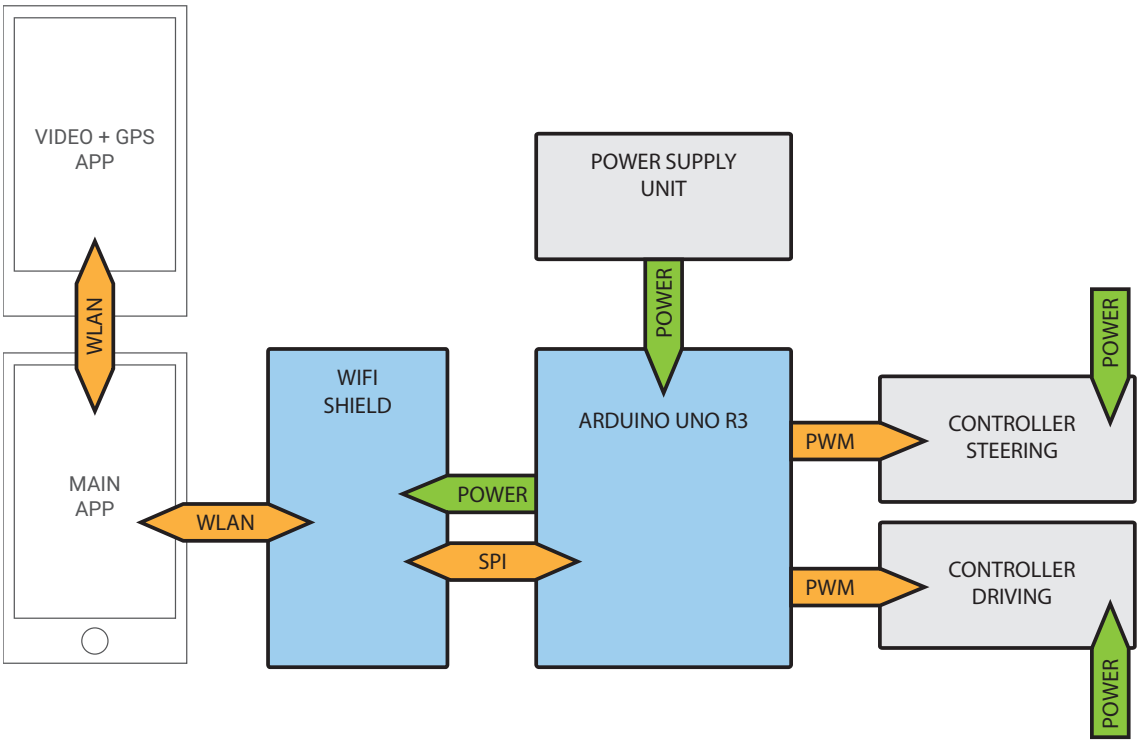

Figure 1: Hardware Interface Overview

Figure 1 is a short overview of all the components used in the projects. The orange marked arrows indicate the interfaces between the devices. The green arrows indicate the power lines.

3 Hardware Block and Signal Specification

The following section used to discuss the hardware interface of every unit more in detail. The signals are labeled referenced to the circuit diagram which is discussed in the section 4 'Hardware Design'.

3.1 Power Supply Unit

3.1.1 Description of block and functionality

The power supply unit consist of 8 batteries which are represented as a battery pack. The voltage is about 9.6 [V] and the capacity is 2000 [mAh].

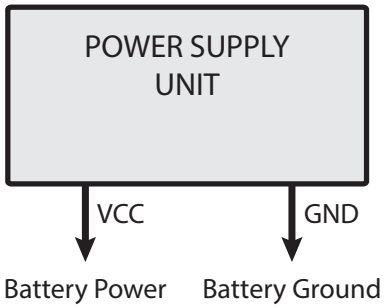

Figure 2: Hardware Interface - *Power Supply Unit*

3.1.2 Signal levels and connections

Signal Name	Interfacing with	Port type	Voltage [V]	Current [mA]	Frequency [kHz]	Remarks
Battery Power	Arduino & Motor-Controller	Output Power	9.6	[0;1000]	DC	VCC for the system
Battery Ground	Arduino & Motor-Controller	Output Power	0	[0;1000]	DC	GND for the system

Table 3: Signal description - *Power Supply Unit*

3.2 Arduino UNO R3

3.2.1 Description of block and functionality

The Arduino is the main control unit of our system. It is used to communicate with the Arduino Wifi Shield and with the motor controllers. The communication between the Arduino Uno and the Wifi Shield is based on SPI. The motor controllers are controlled with an PWM Signal.

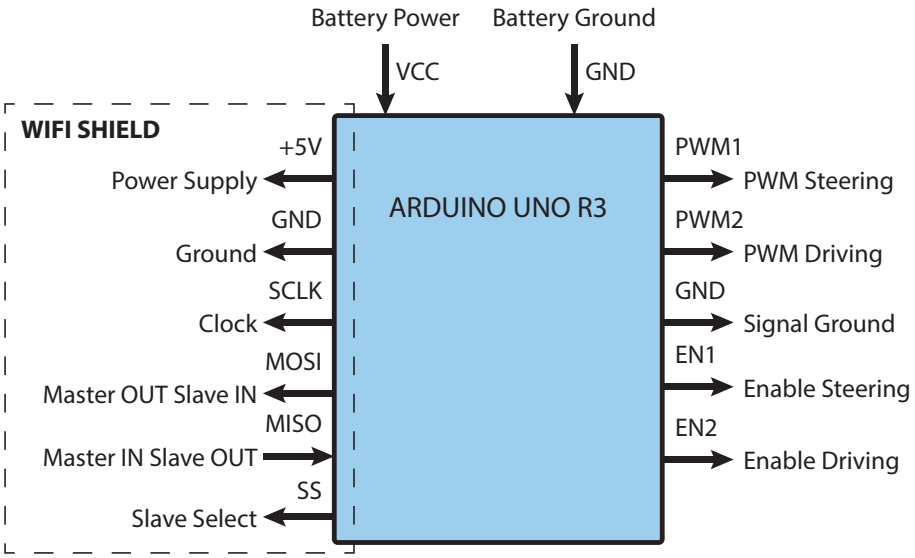

Figure 3: Hardware Interface - *Arduino UNO R3*

3.2.2 Signal levels and connections

Signal Name	Interfacing with	Port type	Voltage [V]	Frequency [kHz]	Remarks
Power Supply	WIFI Shield	Output	5	DC	**VCC** - for the Shield
Ground	WIFI Shield	Output	0	DC	**GND** - for the Shield
Clock	WIFI Shield	Output	[0;5]	DC	**CLK** - Clock Signal for SPI
MOSI	WIFI Shield	Output	[0;5]	DC	**MOSI** - Master Out Slave In Data
MISO	WIFI Shield	Input	[0;5]	DC	**MISO** - Master In Slave Out Data
Slave Select	WIFI Shield	Output	5	DC	**SS** - Slave Select
PWM Steering	Controller 1	Output	[0;5]	DC	**PWM1** - Steering
PWM Driving	Controller 2	Output	[0;5]	DC	**PWM2** - Driving
GND	Controller	Output	0	DC	**GND** - Ground
Enable 1	Controller 1	Output	[0;5]	DC	**EN1** - Enable Controller Steering
Enable 2	Controller 2	Output	[0;5]	DC	**EN2** - Enable Controller Driving

Table 4: Signal description - *Arduino UNO*

3.3 Wifi Shield

3.3.1 Description of block and functionality

The wifi shield is used to communicate with the smartphones over WLAN. The pin headers of the shield fit exactly with the Arduino Uno. Therefore it is very easy to install it. The communication between these two devices is based on an SPI Bus system. Hence there are several pins used for this bus. This have to take in consideration by designing the interface for other devices.

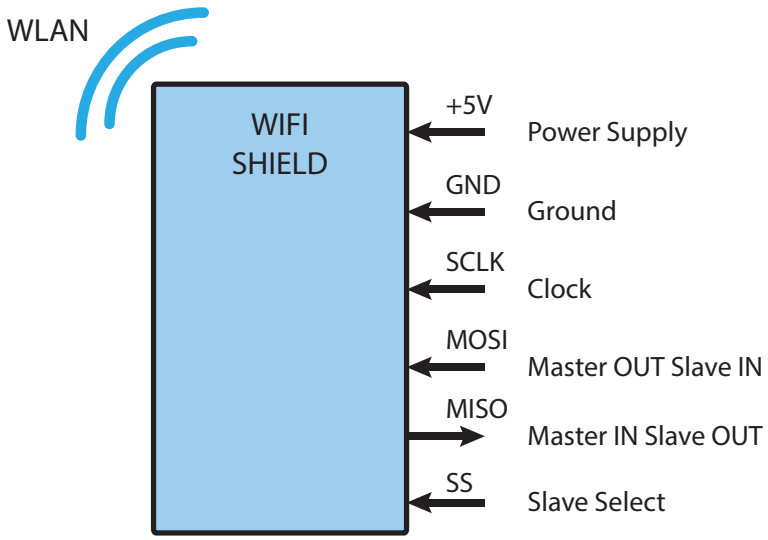

Figure 4: Hardware Interface - *Arduino Wifi Shield*

3.3.2 Signal levels and connections

Signal Name	Interfacing with	Port type	Voltage [V]	Frequency [kHz]	Remarks
Power Supply	Arduino UNO	Input	5	DC	**VCC** - for the Shield
Ground	Arduino UNO	Input	0	DC	**GND** - for the Shield
Clock	Arduino UNO	Input	[0;5]	DC	**CLK** - Clock Signal for SPI
MOSI	Arduino UNO	Input	[0;5]	DC	**MOSI** - Master Out Slave In Data
MISO	Arduino UNO	Output	[0;5]	DC	**MISO** - Master In Slave Out Data
Slave Select	Arduino UNO	Input	5	DC	**SS** - Slave Select
WLAN	Main Smartphone	Bidirectional	5	DC	**WLAN** - Wireless Interface

Table 5: Signal description - *Arduino Wifi Shield*

3.4 Motor Controllers

The devices are developed for first semester students at the Aarhus University. They use it also for a remote controlled car. Hence it is the perfect fitting product for our case.

3.4.1 Description of block and functionality

The controllers are used to generate a power output signal based on the PWM input signal. PWM - *Pulse-Width Modulation* is a very common way to control motors. The PWM Signal is described in the next subsection more in detail. An enable contact is used to enable the PWM generation.

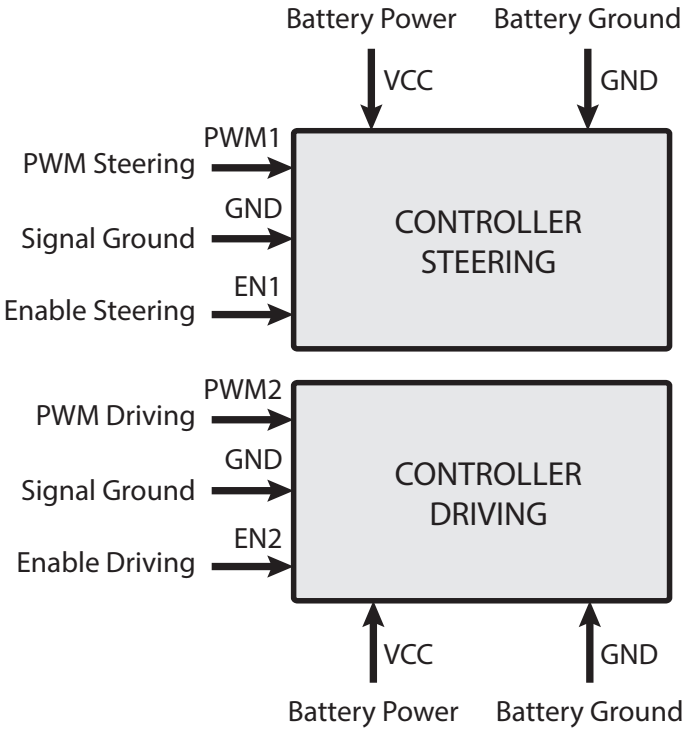

Figure 5: Hardware Interface - *Motor Controllers*

Figure 5 shows also the power supply connections these connections are direct gathered from the battery pack which is described in section 3.1

3.4.2 Signal levels and connections

Signal Name	Interfacing with	Port type	Voltage [V]	Frequency [kHz]	Remarks
Power Supply	Battery	Input	9.6	DC	**VCC** - for the controllers
Ground	Battery	Input	0	DC	**GND** - for the controllers
PWM Steering	Arduino Uno	Input	[0;5]	DC	**PWM1** - Steering
PWM Driving	Arduino Uno	Input	[0;5]	DC	**PWM2** - Driving
GND	Arduino Uno	Input	0	DC	**GND** - Ground
Enable 1	Arduino Uno	Input	[0;5]	DC	**EN1** - Enable Controller Steering
Enable 2	Arduino Uno	Input	[0;5]	DC	**EN2** - Enable Controller Driving

Table 6: Signal description - *Motor Controllers*

3.5 Wifi Communication

3.5.1 Description of block and functionality

The main smart phone interact with the videostream phone and the arduino shield with wireless lan. On the main smartphone is a wifi hotspot enabled. Hence the videophone and the Arduino shield can connect to this hotspot and communicate with the installed TCP server on the main application.

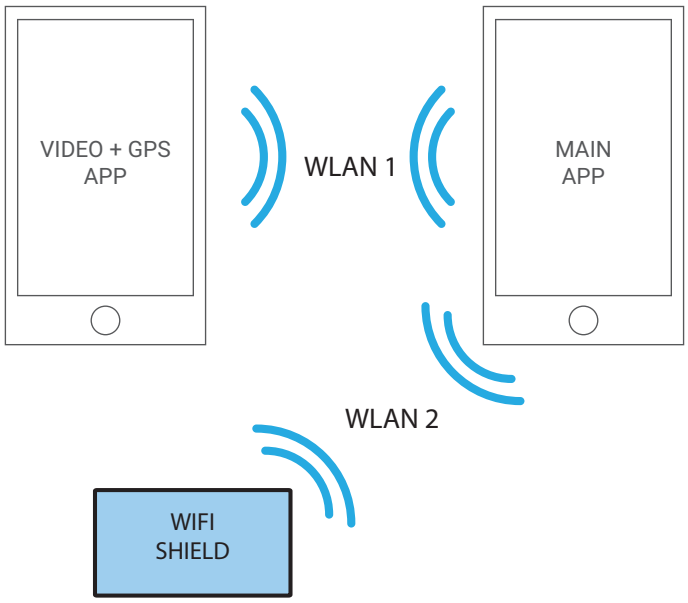

Figure 6: Hardware Interface - *WIFI Communication*

3.5.2 Signal levels and connections

In the following table is the perspective from the main application phone to other devices.

Signal Name	Interfacing with	Port type	Remarks
WLAN 1	Video Stream Phone	Bidirectional	**WLAN** - Wireless Interface Video
WLAN 2	Arduino Wifi Shield	Bidirectional	**WLAN** - Wireless Interface Arduino

Table 7: Signal description - *Wifi Communication*

4 Hardware Design

4.1 Hardware Design Overview

In this section we will give an overview how the devices are connected togehter. After this we will discuss the main hardware components more in detail. We will describe also the interface techniques more in detail.

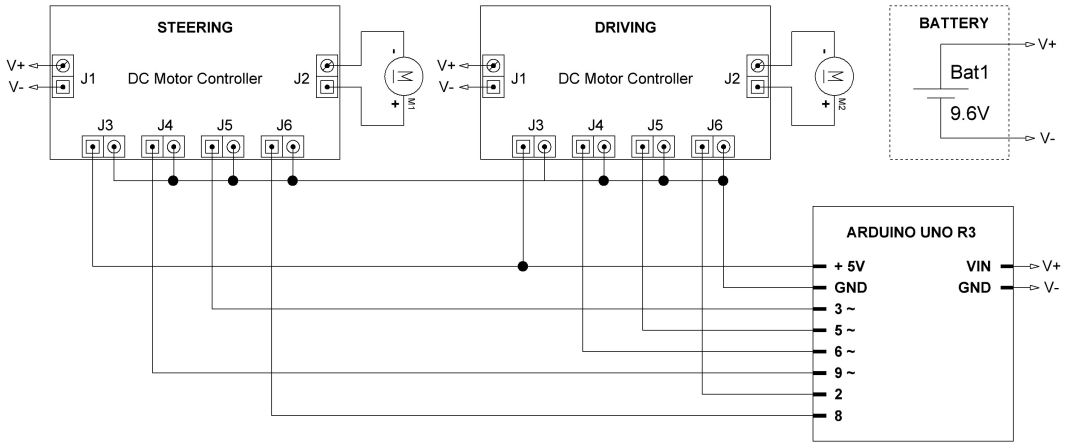

Figure 7: Schematic - *Overview*

Figure 7 shows the overview circuit plan of the car controller.

4.2 Motor Controllers

The motors what we need to control are normal DC motors. Hence it allows us to control them by an simple PWM signal provided from the Arduino UNO board.

4.2.1 Circuit Diagram

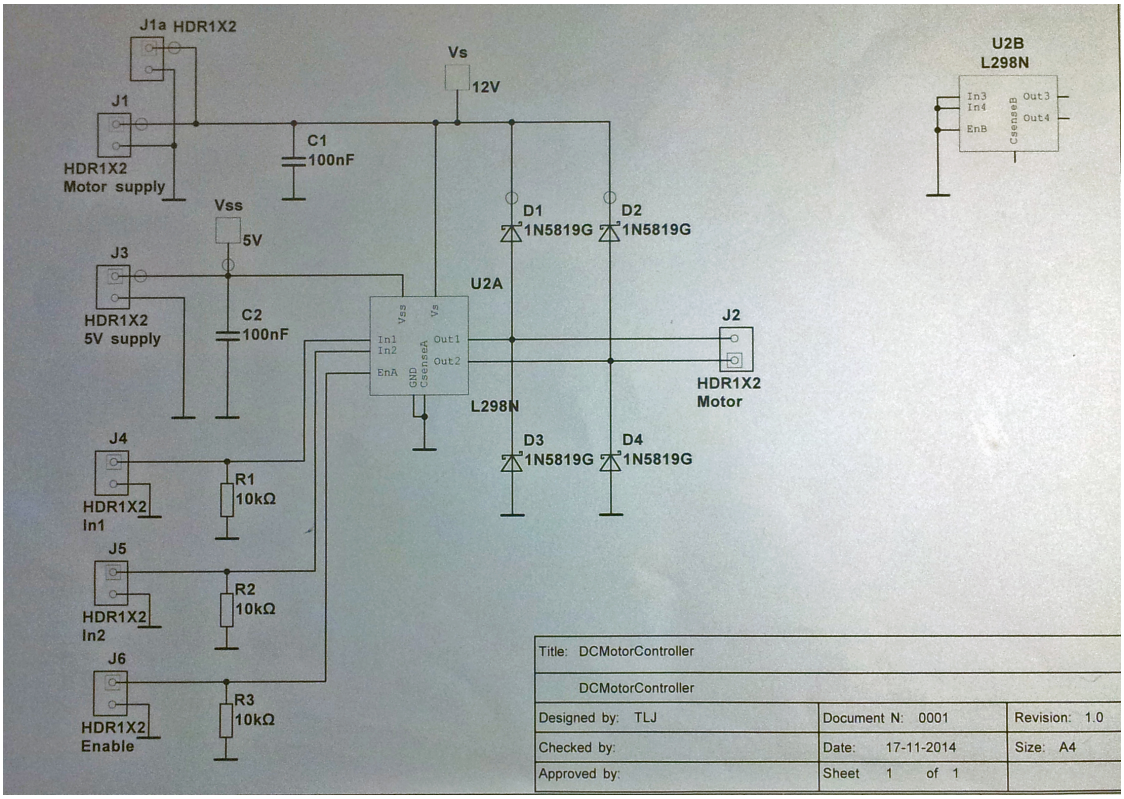

Figure 8: Schematic - *Motor Controllers* (*Source: Aarhus University*)

4.2.2 PWM - Speed Controlling

PWM - *Pulse-Width Modulation* is a very common used technique to control the speed of motors. The aim is to reduce the average voltage of the output signal. The output signal has a maximum voltage and a duty cycle. A duty cycle is the relationship between the ON (**t1** in figure 9) time and period (**T** in figure 9). The duty cycle can be calculated with the following equation:

$$\frac{t1}{T} = duty\,cycle \tag{4.1}$$

The duty cycle is represented as an percent value.

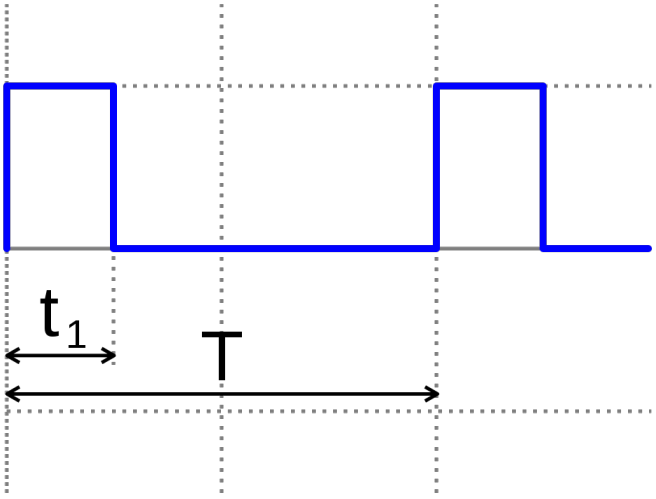

Figure 9: PWM Specification (*Source: www.wikipedia.org*)

In our case the **t1** has a resolution of 8 bits. Hence there are 255 different states to control the motor. Whereas 0 equals no output signal 255 equals full output voltage. By default the Arduino board has a PWM frequency of 490 [Hz]. Hence the period time is 2 [ms].

4.3 Arudino Uno

Arduino is a very easy programmable micro controller. Hence prototyping is much faster by using Arduino Products. We use the 3. Revision of the shield. Hence everything is referenced to this Revision.

4.3.1 General Features

- Microcontroller: ATmega328
- Operating Voltage 5V
- Input Voltage (recommended) 7-12V
- Input Voltage (limits) 6-20V
- Digital I/O Pins 14 (of which 6 provide PWM output)
- Analog Input Pins 6
- DC Current per I/O Pin 40 mA
- DC Current for 3.3V Pin 50 mA
- Flash Memory 32 KB (ATmega328) of which 0.5 KB used by bootloader
- SRAM 2 KB (ATmega328)

- EEPROM 1 KB (ATmega328)
- Clock Speed 16 Mhz

Text Reference: [1]

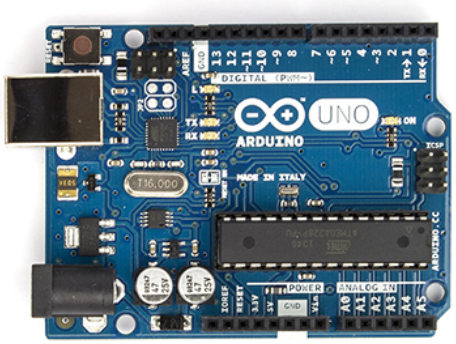

Figure 10: Arduino Uno Board (*Source: www.arduino.cc*)

4.3.2 Programming the shield

The programmer is directly on-board which makes it really nice handle. All you need to start is the board and a USB cable to connect to the board. The controller is also powered by the USB connection. Arduino has his own Software which makes it very easy to program. There are a lot of implemented library's which can be used easily.

4.3.3 Debugging

Debugging direct on the board is not possible. The Arduino IDE doesn't support an Emulator. Hence it is only possible to 'debug' the system with the RS232 interface for communicate with the PC.

4.4 Arduino WIFI Shield

With the Arduino WIFI Shield it is possible to create your own Network. The shield is directly connected with the Arduino Uno with an SPI bus system. The module supports TCP or UDP as communication protocol. There it is possible to choose between TCP Server or Client. In our case on this device runs a TCP Client.

4.4.1 General Features

- Operating voltage 5V (is supplied from the Arduino Uno Board)
- Connection via: 802.11b/g networks
- Encryption types: WEP and WPA2 Personal
- Connection with Arduino on SPI port on-board micro SD slot
- ICSP headers
- FTDI connection for serial debugging of WiFi shield
- Mini-USB for updating WiFi shield firmware

Text Reference: [1]

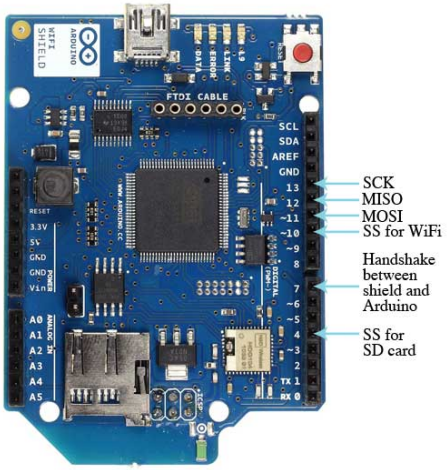

Figure 11: WIFI Shield Interface (*Source: www.arduino.cc*)

In figure 11 you are able to see these pins who are used for the communication. Hence this we had to take in consideration by designing the interface for the motor controllers.

4.4.2 SPI Bus

Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. It can also be used for communication between two microcontrollers.

Figure 12: SPI Interface (*Source: www.wikipedia.org*)

With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices. Typically there are three lines common to all the devices:

- **MISO (Master In Slave Out)** - The Slave line for sending data to the master,
- **MOSI (Master Out Slave In)** - The Master line for sending data to the peripherals,
- **SCK (Serial Clock)** - The clock pulses which synchronize data transmission generated by the master and one line specific for every device:
- **SS (Slave Select)** - the pin on each device that the master can use to enable and disable specific devices.

When a device's Slave Select pin is low, it communicates with the master. When it's high, it ignores the master. This allows you to have multiple SPI devices sharing the same MISO, MOSI, and CLK lines. *Text Reference:* [1]

References

[1] Arduino. (2014). Official website, Arduino, [Online]. Available: `http://arduino.cc/` (visited on 11/30/2014).

List of Figures

List of Tables