Digital Project
Digital Camera Interface

Iñaki Navarro Oiza

May 2004

Abstract

This document describes the development of a prototype of an interface between a CMOS camera and a computer. This interface allows a user to get images from the camera, to change some of the properties of the camera as brightness, luminance, etc. In addition some image process is implemented allowing the camera to track white objects and follow them with a servomotor. The interface was implemented using the Atmel AVR ATmega16 microcontroller.

This document is a final report of the course Digital Project of LTH (Lunds Tekniska hogskola) Sweden. This report and information related to this project can be found in http://www.robozes.com/inaki/dproject

Contents

Chapter 1

Introduction

1.1 Background

For the last four years I have been designing and building mobile robots based in microcontrollers. In this course on Digital Project I wanted to do something about robotics, but developing a robot would have been very expensive and hard in the mechanical aspects. I decided to work with something related with robotics but cheaper and in someway easier: the interface with a digital camera that later I could add to one of my robots.

I own a the digital camera C3088[1]. This camera is used in the CMUcam[2] interface to take images, process them and extract some interesting features that can be used by a robot. My idea was to reproduce the similar characteristics of the CMUcam using the C3088 camera. I wanted that my interface would be able to get images from the camera, change the different settings of the camera (luminance, brightness, etc.), and if possible make some processing of the image (like detecting colors or lines), in a similar way as the CMUcam does.

Also I had the opportunity to use the AVR ATmega16[3] microcontoller and learn about it. I decided to work with it despite it was the first time it was used in the course.

1.2 Specifications

The aim of the project is the design and construction of an interface between the CMOS camera c3088 and a computer, using the AVR microcontroller. The information flows in two ways: on the one hand there are commands from the computer to the camera to change different characteristics of it, on the other images from the camera should be sent to the computer. The communication between the computer and the AVR is through the serial port. The communication between the camera and the microcontroller is: using the I2C protocol to access to the different registers of the camera; and using an 8 bit port to read the images. In addition the camera will be connected to a TV with its analog output for debugging purpose. In the next diagram of Figure 1.1 we can see the main blocks of the design and how information flows.

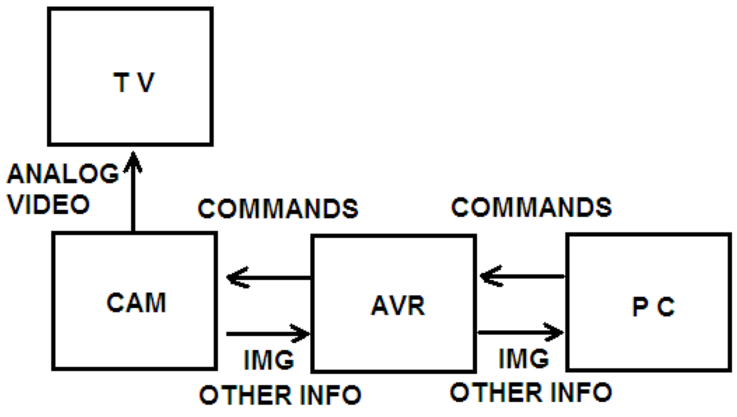

Figure 1.1: Blocks and information flow.

The proposed tasks for the project are the next:

- Make the AVR ATmega16 and its environment work
- Communicate the AVR ATmega16 with the computer
- Make the camera work and see images in the TV
- Read and write the registers of the camera using the I2C protocol
- Read a line from the camera and send it to the computer
- Read a whole image from the camera and send it to the computer.
- Make a little process of the image inside the AVR and send the result to the computer (If there is time and it is possible)

Chapter 2

Description of the system

2.1 The C3088 Camera

The C3088 is the camera used in the project and it is the main part of it, that is the reason that a short explanation is made. It is a color camera module with digital output. It uses a CMOS image sensor OV6620[4] from Omnivision[5]. It has a digital video port that supplies a continuous 8/16 bit-wide image data stream. All the camera functions, such as exposure, gamma, gain, white balance, windowing, can be changed through I2C interface by writing in some registers.

The video output can be expressed in different formats, and with different type of channels (RGB, UVY). The format used in the project is the simplest one: Zoom Video Port Format. In this format 8 bits represent the intensity (Y channel) of one pixel , the other 8 bits represent the U and V channels, but are not used. The information is sent continuously and is synchronized by the HREF, VSYNC and PCLK signals as showed in the next graph in Figure 2.1. The VSYNC signal indicates new frame, the HREF signal indicates when the information is valid and makes the horizontal synchronization, and PCLK is the clock signal (the data is valid in the rising edge). The period of PCLK can be changed by writing in the registers of the camera. This will allow to read images from the camera directly with the microcontroller without the use of additional hardware.

The registers accessed by the I2C bus are 80 and as it was said above allow to change different properties of the camera. In this project they are used to change the period of PCLK, to read the size of the image, to make a mirror of the image, and to reset the camera. It is important here to make a small clarification about the I2C protocol in the camera. The datasheet of the C3088 camera says that the bus used is I2C, but nothing says about the registers and the addresses that should be used. So it is necessary to read the datasheet of the OV6620 chip. There you can find that the protocol used for the communication is not I2C, it is SCCB[6]. If you read the specifications of this protocol you can see that it is for some modes the same as I2C protocol. The ID address of the

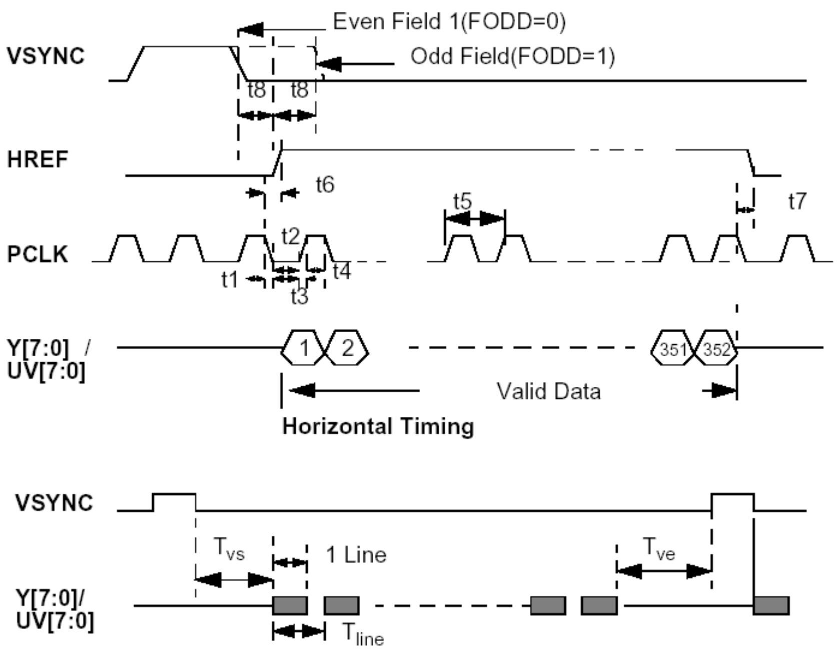

Figure 2.1: Output signals from the C3088.

camera is 0xC0 for writing and 0xC1 for reading, as is said in the datasheet of the OV6620.

The pins of the camera are the next:

- Y0 Y7 Digital output Y Bus.
- UV0-UV7 Digital output UV bus. (Not used)
- PWDN Power down mode and RST Reset
- SDA I2C Serial data and SCL I2C Serial clock input (For the I2C communication)
- FODD Odd Field flag (Not used)
- AGND Analog Ground, GND Common ground and VCC Power Supply 5VDC
- HREF Horizontal window reference output, VSYN Vertical Sync output and PCLK Pixel clock output (For synchronization when getting the images)
- EXCLK External Clock input (not used)
- VTO Video Analog Output (To get images from the TV)

2.2 Hardware

The different modules of the design are explained in this section. The only voltage used is Vcc = 5V. In Appendix A.1 a complete schematic can be found.

2.2.1 Reset and Clock

A typical reset circuit is used with a pull-up resistor of 10Kohms and a switch to ground. Also the reset pin is connected to the JTAG connector for the JTAG ICE[7][3]. A crystal of 16 MHz is connected between the XTAL1 and XTAL2 pins, also two capacitors of 20pF are connected between those pins and ground. This circuits can be seen in Figure 2.2

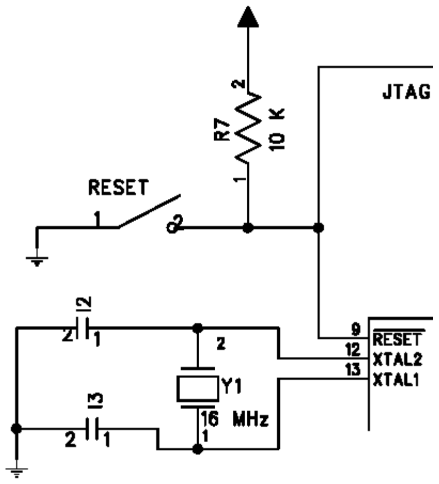

Figure 2.2: Reset and clock circuit.

2.2.2 JTAG ICE

The pins PC2, PC3, PC4 and PC5 are connected to the JTAG ICE connector and also to a pull-up resistor of 10Kohm. RESET, GND and VCC are also in this connector that allow us to program the microcontroller. We can see it in Figure 2.3

2.2.3 LED and switch

In order to do some tests and debug, a LED was connected between PD7 and ground through a resistor of 200 ohms. Also a switch was connected to PD6 with a pull up resistor of 10 Kohm, as it can be seen in the Figure 2.4

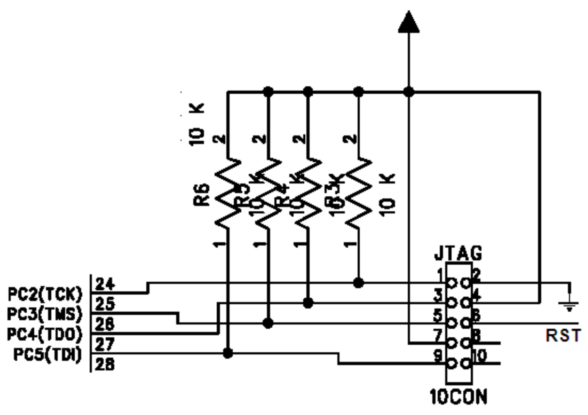

Figure 2.3: JTAG ICE connector.

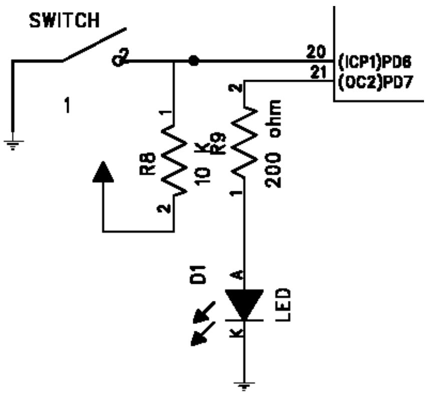

Figure 2.4: LED and switch circuit.

2.2.4 Serial communications

The connection to the serial port of the computer is made using the integrated circuit MAX232[8], because the TTL levels of the microcontroller are not compatible with the computer. PD0 (RXD) and PD1 (TXD) are connected to the MAX232 as it can be seen in the Figure 2.5 The TX and RX pins of the MAX232 with RS232 levels are connected to the computer through a DB9 connector. The capacitors of the circuit are of 1uF.

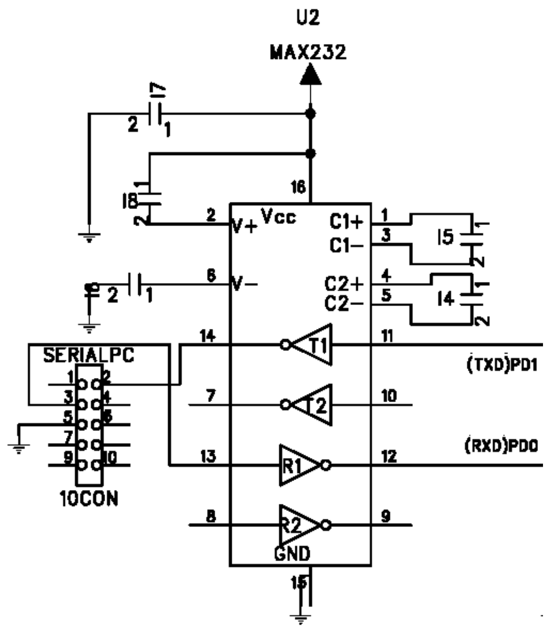

Figure 2.5: Serial Communication circuit

2.2.5 C3088 Camera

The camera operation voltage is as the rest of the circuits 5V, and pins 20 and 22 that are VCC are connected to it. Pin 31 is GND and pins 21 and 15 are AGND (analog ground) and all of them are connected to the common ground. The pins PWDN and RST are connected to ground, so all the resets of the camera are performed by software.

The bus Y0-Y7 of the camera is connected to the port A (PA0-PA7). PCLK is connected to PD2, HREF to PD3 and VSYNC to PD4. The bus UV is not connected because it is not used. SDA and SCL (I2C bus) from the camera are connected to PC1 and PC0 in the microcontroller, because these pins are connected to the TWI hardware that is used to implement the I2C protocol. VTO pin that is the Video Analog Output is connected to the TV through a

RCA connector. Pins FODD and EXCLK are unconnected. In Figure A.1 in Appendix A all these connections can be found.

2.2.6 Servo

To demonstrate a small image process performed with the microcontroller a servo motor is connected and controlled by the AVR. It is connected to pin PD5, because this pin is connected to the comparator of the counter1, that is used to give the position of the servo. Vcc and ground are also present in the connector of the servo.

2.2.7 Comments

The election of which part was connected to which pin was easy. The JTAG should be connected to pins PC2-PC5, because is required by the microcontroller. The I2C bus was implemented using the help of the TWI hardware of the microcontroller, so it should be connected to pins PCO and PC1. The hardware serial port is in pin PD0 and PD1 so that is why these pins were used. For the connection of Y bus of the camera one complete port should be chosen. Ports C and D could not be chosen because they were used for other things. Port A was chosen because the only peripheral that it has is the AD converter and it is not required in this project. PCLK, HREF, VSYN, the servo, the LED and the switch were connected to pins of port D. Port B remains completely free. If in a future we want to use the UV bus to get color images from the camera, it can be connected to the port B of the AVR. Pins PC6 and PC7 are also free.

2.3 Software

All the software of the project was implemented in C. The different parts of the software are explained in this section.
In the web page http://www.robozes.com/inaki/dproject the code of the project can be found.

2.3.1 Serial Communications

The communications with the computer via serial port was the first thing to be implemented because it allows to debug by printing messages in the computer. In the computer side two text terminals were used: Hyperterminal[9] of Windows and RealTerm[10] of the open source community. The serial port settings implemented are the next: 115200 bps, 8 bits, 1 stop bit, 0 parity bits. A velocity of 230400 bps could have been implemented in the microcontroller, but the computer can not work with it.

Because code for the serial port is widely used, the code used was based in the library Atmel AVR USART Library for GCC[11]. The functions from the library were modified to fit the necessities of this project. These functions implement a receiving buffer: the received bytes are read by an interruption and saved in the buffer. This is the only interruption used in the system. To send bytes there is no buffer and the sending functions are blocking. We can find this functions in USART.H and USART.C.

2.3.2 I2C Communication with the camera

The I2C[12] bus is a communication protocol developed by Philips. In this protocol two pins are used, one is the clock and the other is the data. Also this protocol has a Master-Slave architecture. In our case the master is the AVR and the slave the C3088 camera. Registers of the camera can be read or written by the AVR. In the writing operation the master put in the bus the writing address of device, and after that put the address of the register it wants to write, and finally the byte it wants to write in the register. The reading operation is similar: first the master put in the bus the writing address of the device it wants to write, after that the the register address to read from, and then the device reading address. Finally the slave puts in the bus the data requested.

The I2C communication was the most difficult part of the project, because of two main reasons. First of all because I2C protocol is not implemented hardware in the microcontoller used. Second because, as it was said in 2.1, the C3088 camera implements the SCCB protocol that it is almost the same as I2C. Three solutions were though to implement the I2C bus:

1. Use a parallel to I2C hardware converter like PCF8584[13]. This was rejected because it will use at least 10 of the pins of the microcontroller and it will not make the software much easier.
2. Implement by software directly all the protocol.
3. Use the TWI (Two Wire Interface) present in the ATmega16, that is a synchronous bus as the I2C, and that used with some changes can implement the I2C protocol.

The last solution was the one chosen. As in the case of the usart, a library[11] found in Internet was used. To test this library another I2C device was connected to the I2C bus. Once readings and writings were working in this device, the same operations were tried in the camera. The result was that the writing worked, but not the reading. After some investigations with the oscilloscope the problem was detected and solved, it was a timing problem. The read_register and write_register functions are implemented in the I2C_CAM.H and I2C_CAM.C

2.3.3 BMP

To send the images to the computer the BMP format was chosen. The reason was that after the headers only the data information of each pixel is sent. This will allow us to send the image to the computer at the same time is read from the camera. This is very important because then it is not necessary to store the information in another memory, or write a program that will generate the image in the computer. We send in real time the image ready to be displayed in the computer.

The BMP file has the next structure[14]:

- Header. Information about the file. 14 bytes.
- Info Header. Information about the image (size, bits per pixel, type of compression, etc.). 40 bytes
- Palette. It translate each pixel information in its color.
- Data. All the pixel data. In our case each byte represent one pixel(its intensity).

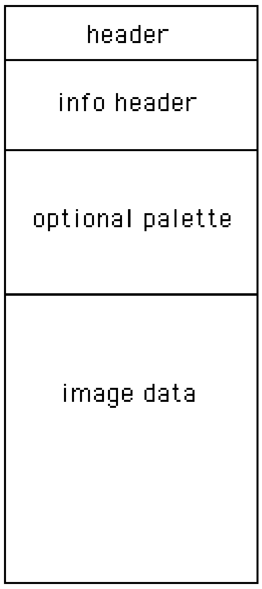

Figure 2.6: BMP file.

Some functions were implemented to create each one of the parts of the bmp file, and can be seen in files BMP.H and BMP.C. createheader and createinfoheader create each one an array of char with the header data. Both header are send before the image through the serial port making use of the function usart_putnumchars. The function sendtable generates and send the palette to the computer. This palette is not stored because it will take to much memory, so it is sent while generated. The last function related to bmp is senddata, that sends to the computer the Data part of the bmp file. In this case It sends a an artificial generated image. When getting and image from the camera and send to the computer this function is substituted by sending the data read from the camera.

2.3.4 Getting Image from the Camera

The Image from the camera is read taking in account the diagram of Figure 2.1. Because the initial frequency of PCLK is 17.73 MHz and the AVR is not fast enough to read each pixel at this frequency two solutions could be taken:

- Use additional hardware to read and store the image.
- Decrease the frequency of PCLK by writing in the register 0x11.

This last solution was the one taken. The frequency taken to read the image depends on the way we read the image. If we read the image by horizontal lines we need to put the lowest frequency allowed that it is: 69,25KHz. This let us to read one line at the same time that is stored in the memory of the AVR. The other mode we read a vertical line of the image in each frame. In this case a higher frequency of 260KHz is used, but we need to read as many frames as vertical lines has an image to get a complete one. In the case of the horizontal lines reading the resulting image is too bright, and that is the reason why the vertical reading is used, even if we need to read as many frames as vertical lines. The horizontal reading is used to read one horizontal line and make a little image process of it. The selection of this frequencies was made experimentally trying to use the highest as possible frequency.

When we want to send an image to the computer the headers and the palette are send to the computer and then we proceed to read the image from the camera. We read from the first frame the first column and send pixel by pixel with the serial port to the computer, after that the second column, and so on until we are done with the whole image. The functions photo and cam_portsinit do this work and can be found in CAM.H and CAM.C.

2.3.5 Little Image ”Process”

The last aim of the project was to do a little process inside AVR. As an example the detection of the horizontal center of a white object was implemented. The way to do it is the next

- An horizontal line is read from the camera and stored in the memory.
- We read each one of the pixel from left to right and see if they are brighter than a threshold.

- Then we look for the zone that being brighter that the threshold is bigger and calculate its center.

This is implemented in the function getcenter in CAM.H and CAM.C.

2.3.6 Servo

To make more visual the getcenter function a servo motor was added to the project to track a white object. A servo motor is a motor with some electronics that is used to set the position of the motor[15]. This position is controlled by the width of a periodic pulse as it can be seen in Figure 2.7. The period is 20ms, and the width of the pulse is between 0.3 ms (to turn it to the left) and 2.3 ms (to turn it to the right). This values are not exactly the same for every servo and must be adjusted experimentally.

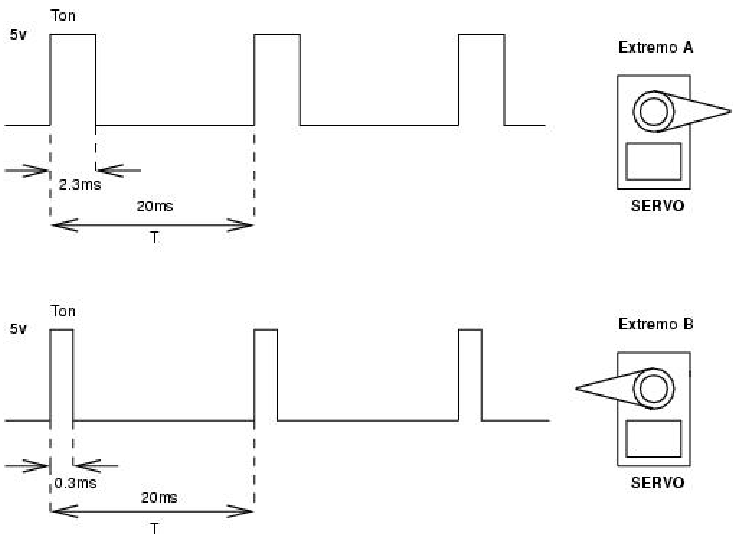

Figure 2.7: Control of a servo.

To generate this periodic signal the Timer1 and its compare output of the ATmega16 were used. The mode of operation used was "Mode 8 PWM, Phase and Frequency Correct" where both period and width can be set independently. To set this mode TCCR1A must be = 0x80 and TCCR1B = 0x12. To get a period of 20 ms then ICR1 = 20000. The width is controlled writing in the register OCR1A.The initialization of the servo is made by the function servo_init, that sets those registers with the appropriate values. To get a and set the position of the servo we use the functions get_servo_pos and set_servo_pos, where the position is given by a value between -900 (left) and 900 (right). The main advantage of using the timer in this mode is that once we set the mode the signal is control generated by hardware and no interruptions are needed. When we want to change the position we only have to call set_servo_pos that will rewrite a new value in OCR1A. All these functions can be found in SERVO.C and SERVO.H.

2.3.7 Main

The main function is implemented in MAIN.C and MAIN.H. This function basically initiates the serial communications, the I2C bus to communicate with the camera, and reset the camera by writing in one of its registers. After that it sends a welcome message to the computer that is printed on the screen. Then goes inside an infinite loop that look for commands sent from the computer and acts according to the command received. This is done by the function readline that reads from the serial port until a buffer is full or until a return character is found. Then these characters read are compared with the different commands implemented. If a command is found then the appropriate action is taken (like read a register and print the result, get an image, move the servo, etc.). All the actions taken are done using the functions explained in the above subsections. After this the loop starts again. If the command is incorrect NCK is printed and the loop starts again.

Chapter 3

Results

The best way to see the results obtained is to show how the system works. Once everything is connected and the terminal program in the computer is running we press the reset button and receive in the screen the next message:

```
....Control Camera Program....
Inaki Navarro Oiza (c)2004
Type HELP and return for help
```

Also on the TV the image from the camera can be seen. If we press HELP and intro we receive the next message:

```
ACK

HELP MENU-Commands:

RR arg1
WR arg1 arg2
READALL
RESET
MIRRORON
MIRROROOFF
PHOTO
TESTBMP
PANORAMIC
SERVO arg1
SCAN
MOVESERVO
TRACK
```

This menu shows the different commands implemented in the system. These commands are explained below:

- RR arg1 reads the register arg1 of the camera. arg1 can be expressed in hexadecimal or decimal format. It prints the result in the form: "Register = XX Value = XX".
- WR arg1 arg2 writes the value arg2 in the register arg1 of the camera. As in the case of RR, arg1 and arg2 can be expressed in hexadecimal or decimal format.
- READALL reads all the registers of the camera and prints them in the same way as RR does.
- RESET resets all the registers of the camera to the preset values.
- MIRRORON makes image to be seen as if it were seen through a mirror, this is the left part of the image is seen in the right part and vice versa. This change can be seen both in the TV and in the images sent to the computer.
- MIRROROFF makes the image to be seen without the mirror property.
- PHOTO sends a BMP file to the computer containing the image seen through the camera. To read the image with the microcontroller the frequency of PCLK is reduced, and for each frame only one column its read. This has as result that an image takes about one minute to be taken and sent at the same time to the computer. This makes that only static images can be taken. The resolution of the image is the resolution of the camera (352x244 pixels). We can see in Figure 3.1 an example image. More images can be seen in Appendix B. To receive the image in the terminal of the computer we should put it in receiving file mode to store it in a file.
- TESTBMP sends a BMP generated by software image to the computer. This image is generated in the microcontroller with few control loops. It can be seen in Figure 3.2
- PANORAMIC takes a panoramic picture using the servo. It first move the servo to the right position and read the center vertical line of the camera and sends it to the computer. After that it moves about 0.1 and read another vertical line. It continues like this 1901 times until it generates a panoramic image of 1901x244 pixels. We can see in Figure 3.3 an example of a panoramic image. More panoramic images can be seen in Appendix B. To get a panoramic image takes about 10 minutes.
- SERVO arg1 moves the servo to the position given by arg1, where -900 means 90 to the left, 0 centered and 900 means 90 to the right.
- SCAN moves the servo to the left and then slowly to the right and again to the left. It is used only to see how can the servo be moved.
- MOVESERVO allows the user to move the servo using the keyboard with the keys P and Q. Pressing space will exit this function.
- TRACK will perform a tracking of a white object with black background. The center of the white object is got with the getcenter function and the servo is moved to the right or to the left to make the object be always in

the center. The tracking was tested with a black page with a white line in the middle an it works pretty well. Pressing space will exit this function.

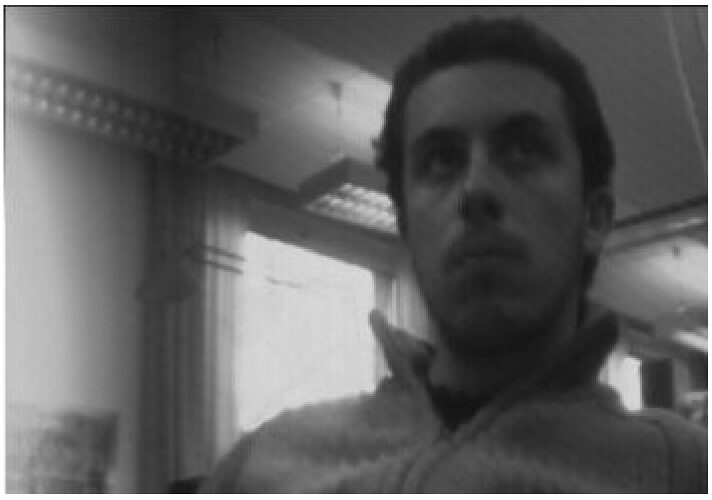

Figure 3.1: Photo taken with the camera

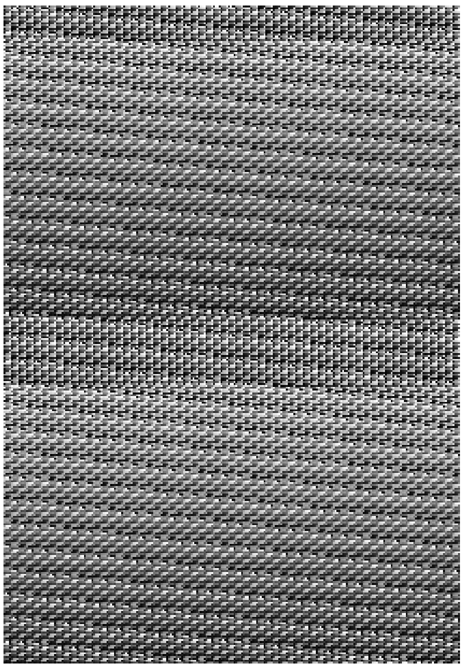

Figure 3.2: Image generated with the AVR

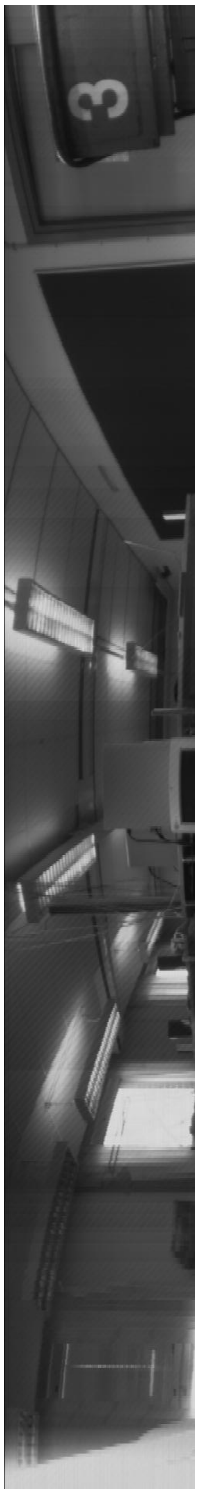

Figure 3.3: Panoramic Photo

3.1 Conclusions

All the proposed tasks of the project were done. The communication with the computer was done, also the I2C communication with the camera. Images can be read and sent to the computer in real time, with the only disadvantage that it takes a lot of time. In addition small ”image process” is implemented by getting the center of a white object. The servo motor was added to the project to show how to track an object using the this information. The servo is also used to get panoramic images.

In Appendix C some photos of the system can be seen.

Appendix A

Schematic

Complete schematic of the system.

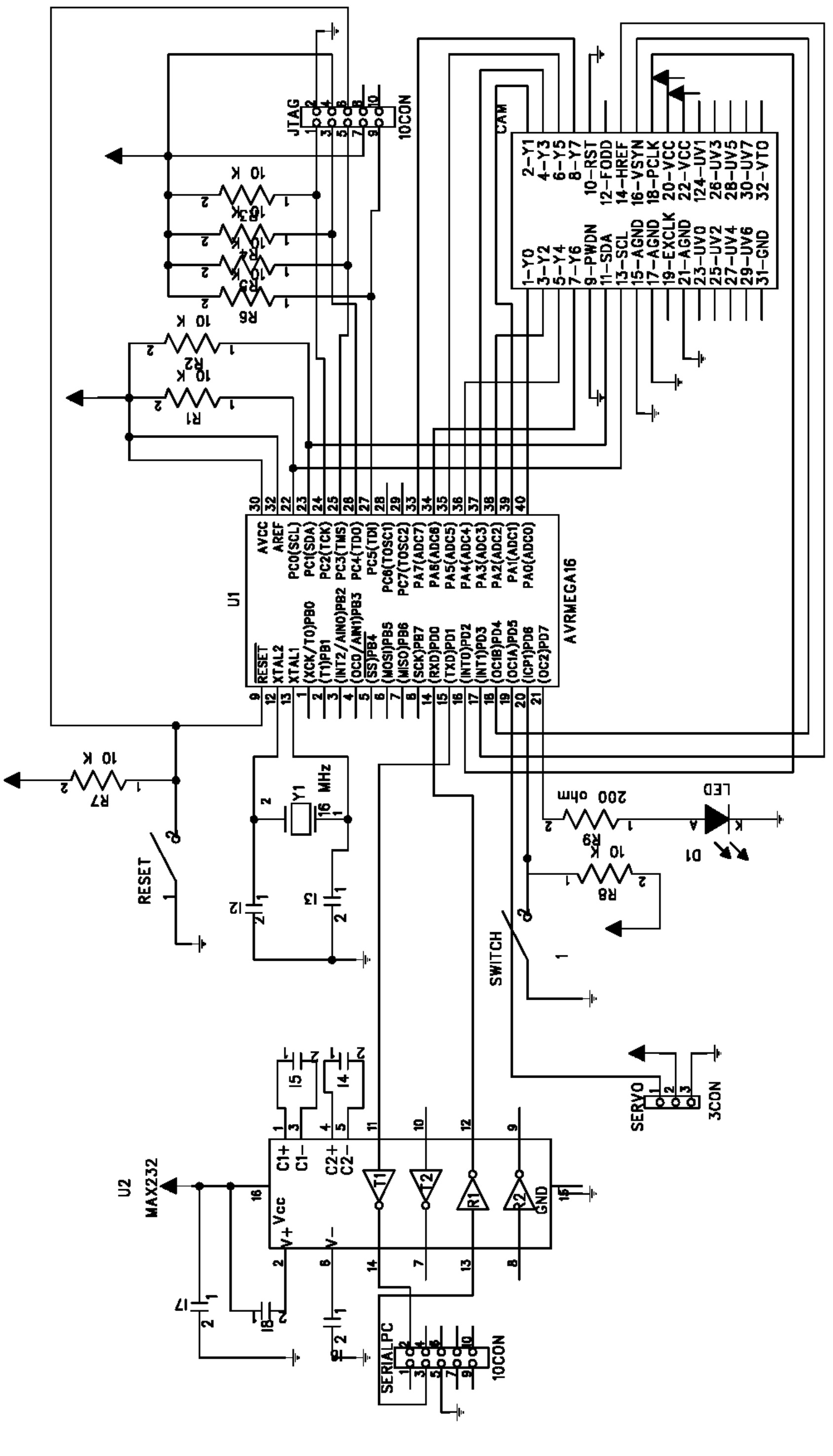

Figure A.1: General Schematic.

Appendix B

Images

Some images taken with the camera and sent to the computer are shown here.

Figure B.1: Photo taken with the camera

Figure B.2: Photo taken with the camera

Figure B.3: Photo taken with the camera

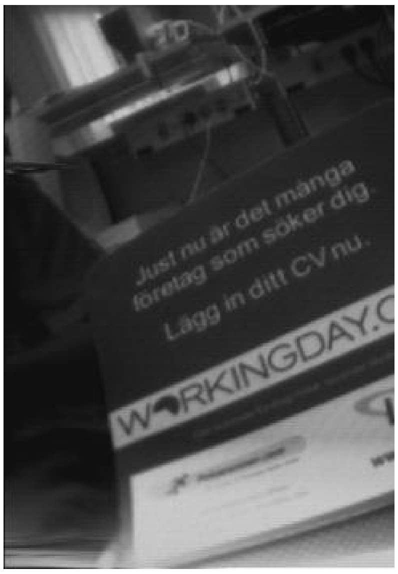

Figure B.4: Photo taken with the camera

Figure B.5: Photo taken with the camera

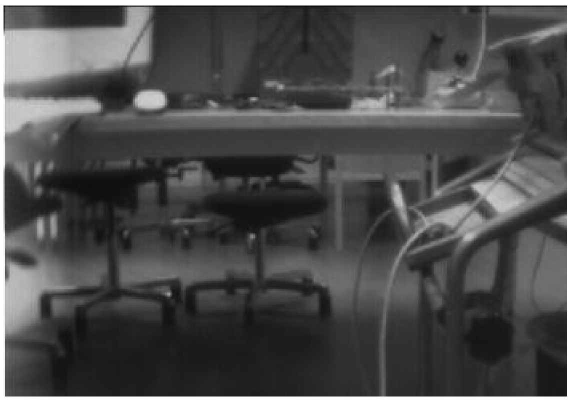

Figure B.6: Photo taken with the camera

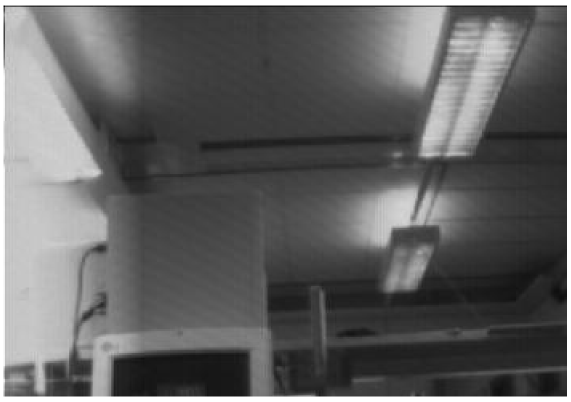

Figure B.7: Photo taken with the camera

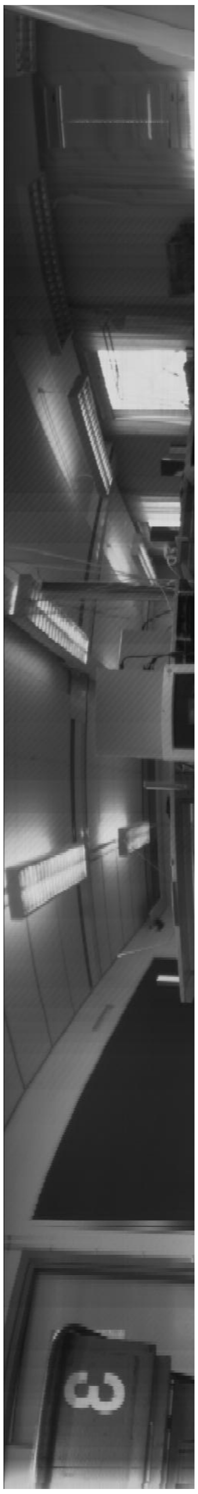

Figure B.8: Panoramic Photo

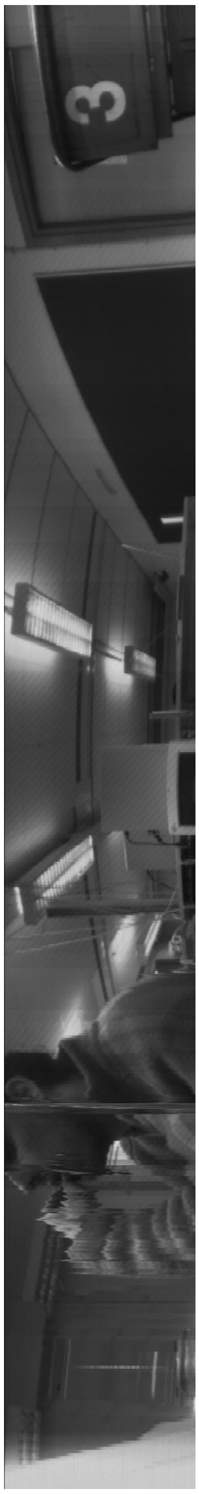

Figure B.9: Panoramic Photo

Appendix C

Photos of the system

Some photos of the system can be seen here.

Figure C.1: c3088 cam

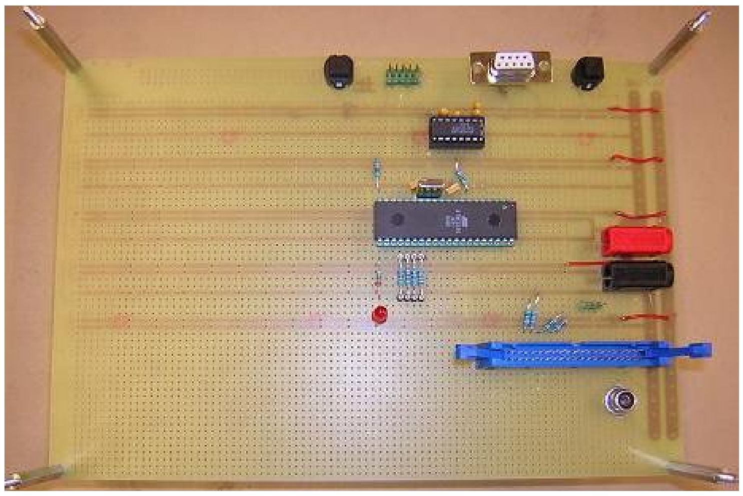

Figure C.2: Electronic board

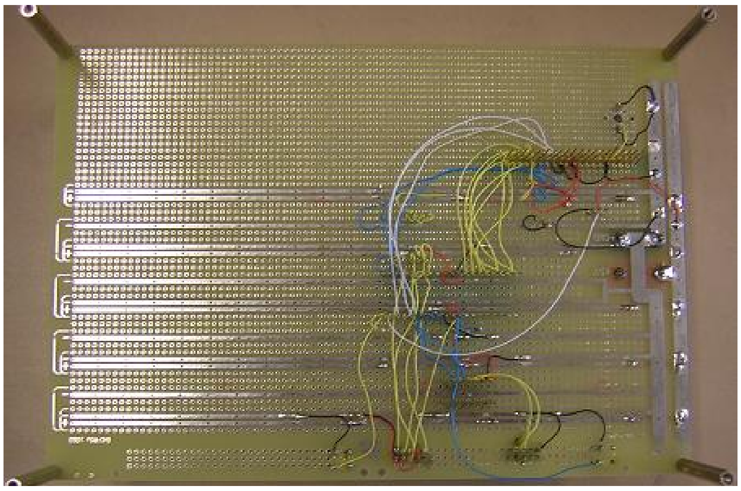

Figure C.3: Electronic board

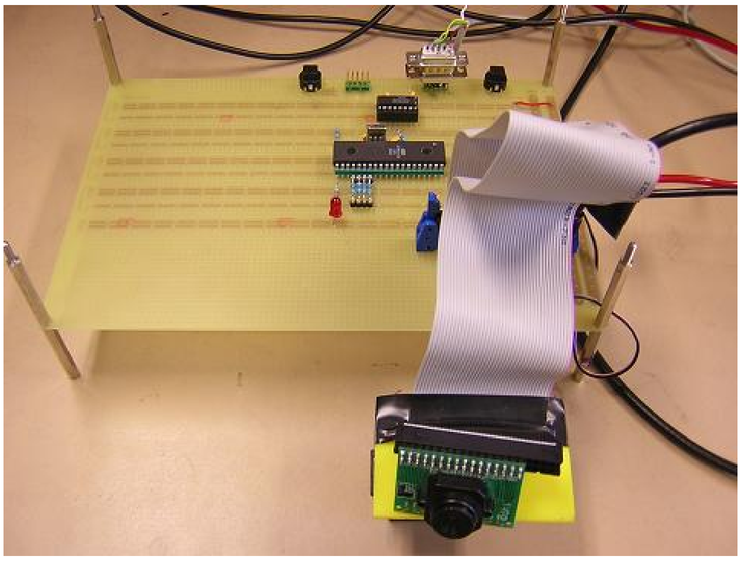

Figure C.4: Complete system

Bibliography

[1] Datasheet of the C3088 Camera
http://www.roboter-teile.de/datasheets/C3088.pdf

[2] Web site of the CMUcam
http://www-2.cs.cmu.edu/ cmucam/

[3] Web site of Atmel with information about AVR ATmega16
http://www.atmel.com/

[4] Datasheet of the ov6620 chip
http://www.ovt.com/cc6620.html

[5] Web site of Omnivision
http://www.ovt.com/

[6] Datasheet of the SCCB bus
http://www.ovt.com/pdfs/ds_note.pdf

[7] Datasheet of the JTAG ICE
http://www.atmel.com/dyn/resources/prod_documents/DOC2475.PDF

[8] Web site of Maxim with info of MAX232
http://dbserv.maxim-ic.com/

[9] Web site of Microsoft with information about Hyperterminal program
http://www.microsoft.com

[10] Web site of the Real Term terminal
http://realterm.sourceforge.net/

[11] Web site of Jaakko Ala-Paavola with libraries for the AVR
http://vodka.tky.hut.fi/ jap/Electronics/AVR/index.html

[12] Web site of the I2C bus
http://www.semiconductors.philips.com/buses/i2c/

[13] Datasheet of the PCF8584
http://www.semiconductors.philips.com/pip/PCF8584P.html

[14] Web page with information about the BMP file (in French)
http://www.progzone.free.fr/graphisme/formats/bmp/bmp.html

[15] Web page with information about servomotors (in Spanish)
http://www.iearobotics.com/proyectos/cuadernos/ct3/ct3.html