People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of

The Requirements for the Degree of

MASTER

In: **Electronics**

Option: Computer Engineering

Title:

FPGA-Based Home Automation System with Embedded-Linux and OpenCV

Presented by:

- SAKEUR Mohamed Elamine

- BELKHIRI Alaaedine

Supervisor:

Dr. MAACHE Ahmed

Registration number:……..../2019

Abstract

Nowadays, home automation has caught the focus of technology. It is the process of connecting and controlling home appliances automatically and/or remotely. In first part of this project, we designed and built an embedded controller on an FPGA to control a small home consisting of one chamber. The system automatically operates the door, the window, the lights, and the alarm. For the second part, we implemented a fire detection system using OpenCV on an Embedded-Linux platform using the same FPGA and a USB webcam. At the moment, both parts are separate and don't operate at the same time. The system uses the Intel DE10 FPGA board and the Qsys platform designer tool to implement both parts. One of the aims of this project is to evaluate the potentials of the newly acquired DE10 FPGA board.

Both parts used the Qsys platform designer to implement the on-chip hardware system inside the FPGA fabric. For the first part, controller software is written in C and executed on the NIOSII soft-core to control the small home. For the second part, embedded Linux is booted from an SD-Card plugged into the DE10. Then, a fire detection algorithm is implemented using OpenCV to capture live video from an attached USB webcam and apply a number of rules to detect the presence of fire.

The first system part was successfully implemented using the Quartus Prime version 16.1, on a DE10 board. Our prototype system includes a USB webcam for fire detection, controlling the opening/closing process of doors/windows, controlling lights and an alarm according to a gas LDRs and IR sensors. Finally, the prototype system worked as expected.

Keywords: FPGA, DE10, home automation, QSYS, VHDL, OpenCV, Cyclone V, Linux, SD-card.

Dedication

I would like to dedicate this modest work to my dear parents, for unterminated encouragement and support. Their excellent guidance made me the person I am today. I would express my gratitude to all my family for their encouragement. I additionally dedicate this project to all my good friends.

M.A.Sakeur

I would like to dedicate this humble work to my beloved parents, for their support and encouragement. I would express my gratitude to all my family for their encouragement and all my good friends for Standing beside me.

A.Belkhiri

Acknowledgment

All praise and thanks giving to Allah the most powerful and most merciful who give us the ability and patience to accomplish this humble work.
We especially thank our supervisor, Dr. MAACHE Ahmed, for having oriented us in the choice of our theme of project and to have guided us throughout this project. We are grateful to our teachers, academic staff and workers at the Institute of Electrical and Electronic Engineering, who prepare for us the environment to work, and offer us their valuable assist.
We would like to acknowledge the donation of the DE10 board done by Intel University Program.
We would additionally like to thank all those people who supported us through the process of writing this report.

Table of Figures

Acronyms

- CPU Central Processing Unit
- DSP Digital Signal Processor
- FPGA Field-programmable Gate Array
- GPIO General purpose Input Output
- IC Integrated Circuit
- IR Infra-red
- JTAG Joint Test Action Group
- LDR Light Dependent Resistor
- LED Light Emitting Diode
- PLL Physical Link Layer
- SD Secure Digital
- SDRAM Synchronous Dynamic Random Access Memory
- SRAM Static Random Access Memory
- SoPC System on Programmable Chip
- UART Universal Asynchronous Receiver Transmitter
- USB Universal Serial Bus
- VHDL VHSIC Hardware Description Language

Introduction

Overview

Nowadays the word ‘smart’ is commonly used, such as smartphone, smart TV, and smart city... The word smart refers to automation and the fact of using new technology to simplify human life.

The notions related to automation of tasks were in existence during long times; now with modern technologies, the automation is easier. FPGA CycloneV on the DE10 represents a good choice. A home automation with a fire detection system is a good experiment for the board.

When it comes to home automation, embedded systems are always involved. FPGA does in particular have a large utilisation in embedded systems field.

FPGAs are simple approach to realize logic circuits, because of their flexibility features. They are Ideal for embedded SOPC (System On Programmable Chip) systems implementation. Where a complete system is designed on a single FPGA chip, and is usually based on a soft-core processor described in general in HDL (Hardware Description Language) [1] .

Figure 1 Home automation system [2]

Objectives

The main objective of this project is to design and implement an embedded system for home automation and security. We aim to control a small home using SOPC on an FPGA. Next, an embedded application based on embedded-Linux will be designed and implemented to detect fire in the house. This application utilises the OpenCV Library to capture and process real-time videos for detecting the presence of fire.

Organization of the Report

This report is divided into four chapters, theoretical background, Home Automation System Design; OpenCV based fire detection and a conclusion. The theoretical background chapter contains an overview on home automation, DE10 board, FPGA, and OpenCV. It also outlines basic concepts about the hardware utilised. Second chapter outlines the details of the design and implementation of the home automation system. Chapter three discusses an OpenCV-based fire detection system and goes through its hardware and software implementation. The conclusion contains some results and providing some suggestions for predictable further work.

Chapter I

Theoretical background

1.1 What is an FPGA

A Field Programmable Gate Array, as its name suggests, consists of an array of logic blocks (Logic cells) whose the interconnections are programmable alongside SRAM blocks, DSP blocks, PLL blocks and I/O pins. FPGAs support JTAG and some devices can be configured via USB.

Specific applications of FPGAs include digital signal processing, software-defined radio, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation, radio astronomy, metal detection and a growing range of other areas [3].

1.1.1 The DE10 Standard Development Board

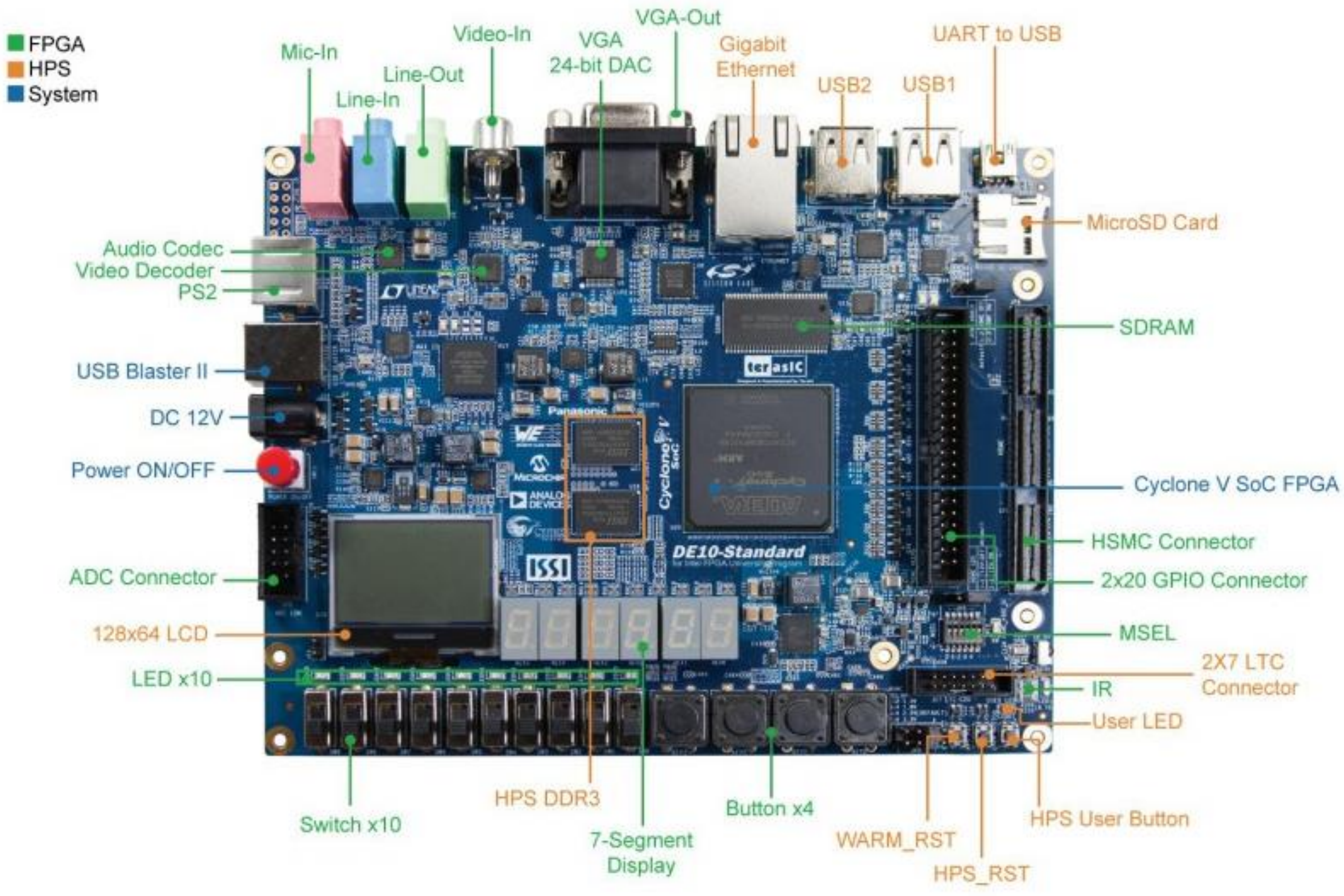


Figure 1.1 DE10-Standard Development Board [4]

The DE10-Standard Development Kit shown in Figure 1.1, presents a robust hardware design platform built around the Intel System-on-Chip (SoC) FPGA, which combines the dual-core Cortex-A9 embedded cores with industry-leading programmable logic for ultimate design flexibility [9].

The DE10-Standard development board includes sophisticated features such as high-speed DDR3 memory, video and audio capabilities, Ethernet networking, and much more. [5] Figure 1.2 demonstrates a DE10 board block diagram.

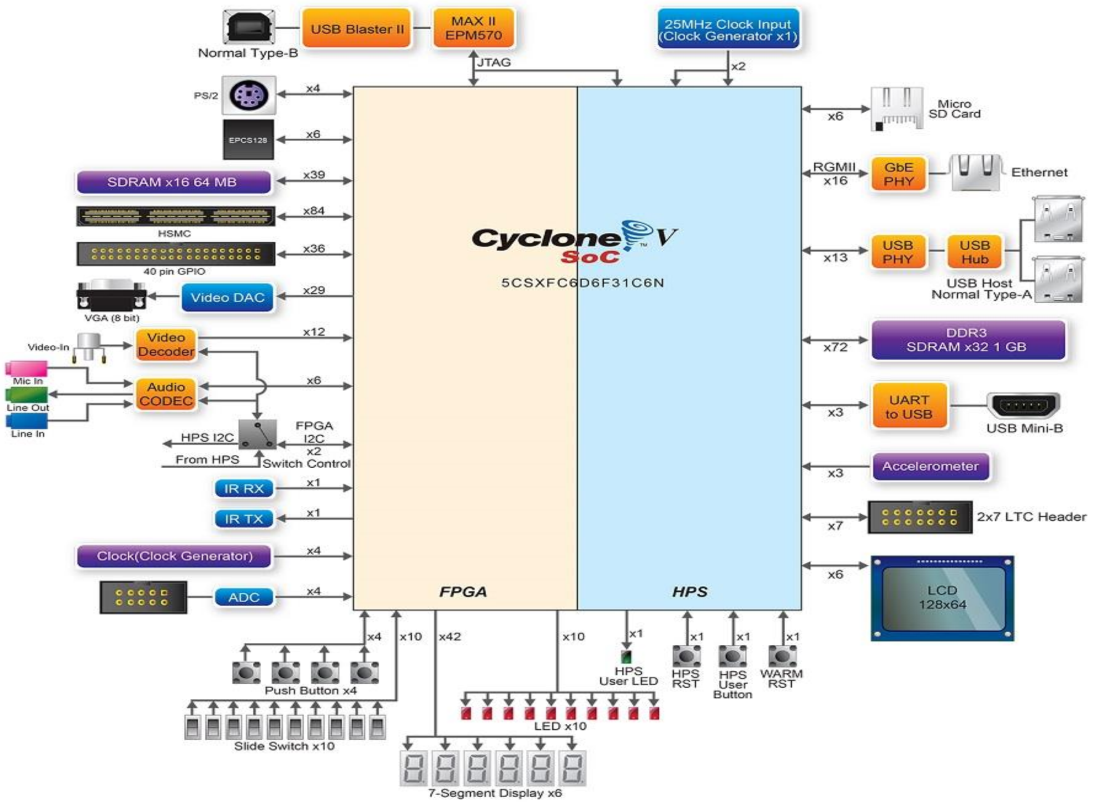


Figure 1.2 DE10-Standard board Block Diagram [5]

A logic cell is composed of one or more LUTs (Lookup tables) made from memory cells (SRAM), and bi-stables (flip-flops). It implements combinational logic as well as sequential logic [3]. FPGA chip adoption is driven by their flexibility, hardware-timed speed and reliability, and parallelism [4].

1.1.2 Overview of Intel Qsys system integration tool

The Qsys platform designer system integration tool in Quartus Prime software is the tool replaces the SOPC Builder tool for new designs, permits faster system development and design reprocess in the FPGA design. Qsys automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems. [5]

1.1.3 Soft-Core Processor

A soft-core processor solution is one that is implemented entirely in the logic primitives of an FPGA using VHDL. Because of this implementation, the Nios 2 processor will not operate at the same clock frequencies or have the same performance of a discrete solution on The DE10 board. In many embedded applications, however, the high performance achieved by the previous two processing options is not required, and performance can be traded for expanded functionality, reduced cost, and flexibility. All the major FPGA vendors have soft-core processors in their products and there are additionally a number of companies and organizations developing soft-core processors that are platform independent and can be implemented in any FPGA design. [6]

It does not have the speed or performance characteristics of a hard-core processor. However, its main features are its expanded functionality and the flexibility of adding a processor to an existing FPGA design or even adding an additional processor to provide more processing capabilities.

The Nios II is a soft-core embedded processor provided by Intel FPGA. It can be connected to other components on the DE10 board to form a complete system implemented in the Cyclone V FPGA chip. These components are interconnected by using the interconnection network called the Avalon Switch Fabric. Figure 1.3 illustrates a Nios II system implemented on the DE10 board.

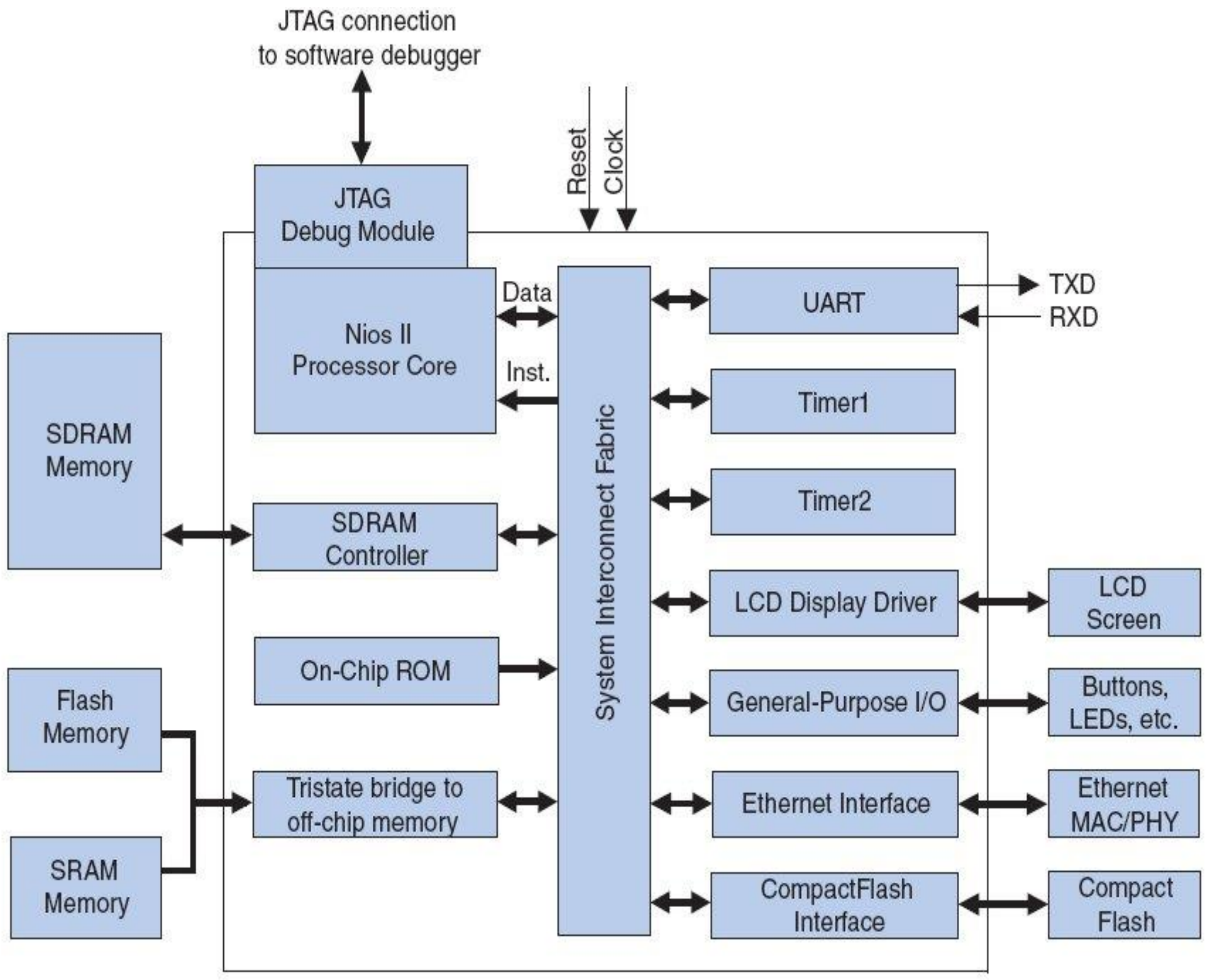
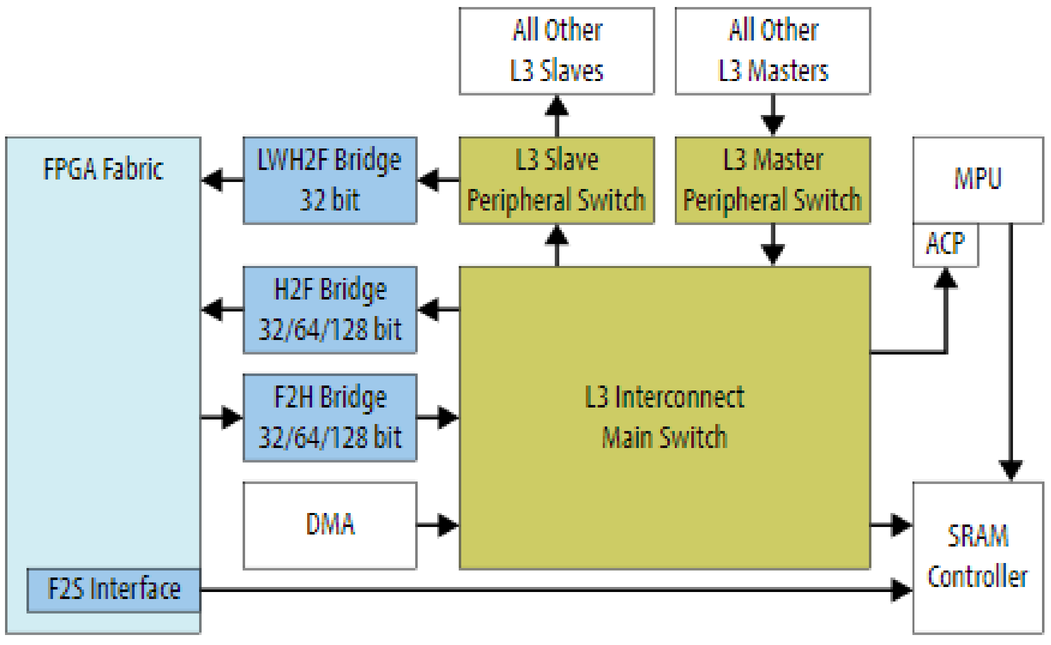


Figure 1.3 A typical Nios II system [30]

1.1.4 Hard processor system

Intel's SoPC integrates an ARM hard processor system (HPS) consisting of processor, peripherals and memory interfaces tied seamlessly with the FPGA fabric using a high-bandwidth interconnect backbone [5] Figure 1.4 shows the HPS-FPGA Bridges. The HPS has three bridges that use memory-mapped interfaces to the FPGA based on the Arm Advanced Microcontroller Bus Architecture (AMBA) and Advanced extensible Interface (AXI) to allow communication between the FPGA and the ARM processor [7].



Key:
H2F: HPS-to-FPGA
LWH2F: Lightweight HPS-to-FPGA
F2H: FPGA-to-HPS
F2S: FPGA-to-SDRAM

Figure 1.4 HPS-FPGA Bridges [7]

1.2 OpenCV

OpenCV is an open source computer vision library. Its overall goal is to create vision to computer. The library is written in C and C++ and runs under Linux, Windows and Mac OS X. There is active development on interfaces for Python, Ruby, Matlab, and other languages. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. OpenCV is written in optimized C and can take advantage of multicore processors [8].

OpenCV has more than 2500 optimized algorithms and it is well suited for computer vision. Images can be represented with different data types such as 8-bit or 16-bit or 32-bit integers, and floating points values in both single (32-bit) and double precision (64-bit) [9].

One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that assists people build fairly sophisticated vision applications quickly. The OpenCV library contains over 500 functions that span many areas in vision, including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision, and robotics. Because computer vision and machine learning often walk hand-in-hand, OpenCV additionally, contains a full, general-purpose Machine Learning Library (MLL). This sub library is focused on statistical pattern recognition and clustering. [8]

RGB colour model and fire detection rules

There are three different element of colour for each pixel: R, G and B. The pixel colour can be extracted into these three individual elements [10].

RGB colour model is used to detect red colour information in image. In terms of RGB values, the corresponding inter-relation between R, G and B colour channels: R>G and G>B.

The combined condition for the captured image can be written as: R>G>B. In fire colour detection R should be more stressed then the other component, and hence R becomes the domination colour channel in an RGB image for fire. This imposes the condition for R as to be over some predetermined threshold value RTH [10].

All of these conditions for fire colour in image are summarized as following:

$$\text{Condition1: R} > RTH \qquad 1$$

$$\text{Condition2: R} > G > B \qquad 2$$

Where 'RTH' is the red colour threshold value for fire [10].

Y, Cb and Cr colour model

Y, Cb and Cr colour space is used in our model rather than other colour spaces because of its ability to distinguish luminance information from chrominance information more effectively than other colour model.

In order to create Y, Cb, Cr components from obtained RGB Image. We will use colour space transformation equation to transform each RGB pixel in corresponding Y Channel, Cb Channel, Cr Channel pixel to form a corresponding Y, Cb, Cr image. When the image is converted from RGB to YCbCr colour space, intensity and chrominance is easily discriminated.

In YCbCr colour space, Y′ is the luma component (the "black and white" or achromatic portion of the image) and Cb and Cr are the blue-difference and red-difference chrominance components, will be chosen intentionally because of its ability to separate illumination information from chrominance more effectively than the other colour spaces.

1.2.1 FPGAs and Computer Vision

It is easy to perceive why Field Programmable Gate Arrays (FPGAs) popularity are getting more and more in the field of Computer Vision. It can potentially speed up image processing by orders of magnitude given its true parallel architecture. However, it comes at a price. It is time consuming to create a design for an FPGA that can process image data satisfactory and donate the same result as, for example, functions from the popular OpenCV (Open Computer Vision) library. OpenCV contains any functions and data structures to aid in C++ computer vision design. Algorithms such as the Harris corner and edge detector or stereo matching are included as functions in OpenCV. Image filters are often the building blocks of such an algorithm. Image filters represent mathematical functions applied to an image using convolution or correlation [9].

1.3 Linux

Operating systems is the first computer program that the computer executes when it is turned on. The operating system manages the communication between the software and the hardware of the computer.

Linux has a various number of distributions where the most popular ones are: Ubuntu Linux, Linux Mint, Arch Linux, Deepin, Fedora, Debian and openSUSE [11] . It is usually composed principally of a Bootloader, the kernel, the Daemons and the shell etc.

The kernel is the "lowest" level of the OS. It is the one piece of the entire that is actually called "Linux". The kernel is the core of the system and manages the CPU, memory, and peripheral devices. [12].

1.4 Home automation

Recently, the notion of the home automation system is becoming an important issue in many publications and home appliances companies. Home automation is a house or living environment that contains the technology to allow devices and systems to be controlled automatically. Remote control is useful to preserve home comfortable and to support the elderly and the disabled people [2]. Figure 1.5 is an illustrative image of Home automation contains some common symbols.

Figure 1.5 a home automation illustrative image [13]

The hardware used in our project consists of:

1.4.1 Sensors

Sensors are analogue devices, which can convert a physical measure to a voltage range.

Gas sensor

In current technology scenario, monitoring of gases produced is very important. From home appliances such as air conditioners to electric chimneys and safety systems at industries monitoring of gases is very crucial. Gas sensors are very important part of such systems.

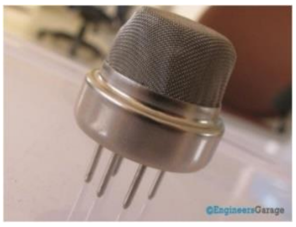


Figure 1.6 Typical Gas Sensor [14]

Gas sensors spontaneously react to the gas present, thus preserving the system updated about any alterations that occur in the concentration of molecules at gaseous state. Figure 1.6 shows atypical gas sensor.

LDR sensor

It is called: LDR, photo resistor, photocell or photoconductor. Light dependent resistors are light sensors used to sense and measure the light intensity. Sensitivity of LDR varies with the wavelength of the light applied to it, its resistance increase as the light intensity decrease, which affect the passing electric current [14]. Figure 1.7 is a graph representing the relation between light intensity and the resistance of LDR.

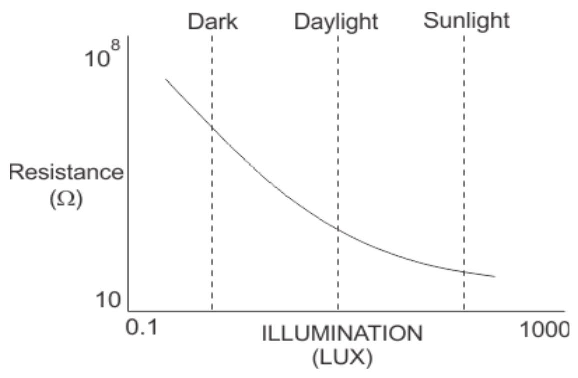


Figure 1.7 The relation between light intensity and the resistance of LDR [15]

IR sensors:

Infrared technology is widely used in wireless applications; it is divided into three regions:

•Near infrared region: from 700 nm to 1400 nm, and are IR sensors, fibre optic.

•Mid infrared region: ranges from 1400 nm to 3000 nm, such as Heat sensing.

•Far infrared region: from 3000 nm to 1 mm, like Thermal imaging.

An infrared sensor is an electronic device used to in sensing motion and remote controls by either emitting and/or detecting infrared radiation. Its waves are not visible to the human eye bounded by the visible and microwave regions.

The purpose of operation of the Infrared Sensor used in this project is to detect if a person crosses through the door. The operation commences by transmitting an infrared signal from an IR emitter. Then, this signal should be received by the IR receiver in case there is no obstacle in between [15].

1.4.2 Actuators

Actuators are devices, which converts an input voltage to a physical act. Here are some demonstrations on some of the actuator utilized in this project.

Stepper Motors

Stepper Motor is a brushless electromechanical device which converts the train of electric pulses applied at their excitation windings into precisely defined step-by-step mechanical shaft rotation. The shaft of the motor rotates through a fixed angle for each discrete pulse. This rotation can be linear or angular. It does one step movement for a single pulse input [16].

Stepper motors are DC motors that move in discrete steps. They have multiple coils that are organized in groups called "phases". By energizing each phase in sequence, the motor will rotate one-step at a time. Figure 1.8 shows an ordinary stepper motor.

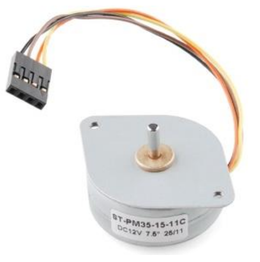


Figure 1.8 an ordinary stepper motor [16]

When a train of pulses is applied, it turns through a certain angle. The angle through which the stepper motor shaft turns for each pulse is referred as the step angle. The number of input pulses given to the motor decides the step angle and hence the position of motor shaft is controlled by controlling the number of pulses.

If the step angle is more diminutive, the greater will be the number of steps per revolutions and higher will be the accuracy of the position obtained. The step angles can be basically as large as 90 degrees and as diminutive as 0.72 degrees, however, the commonly used step angles are 1.8 degrees, 2.5 degrees, 7.5 degrees and 15 degrees [16].

Octal Buffers with 3-state outputs (SN74LS245) are used to isolate the inputs and the outputs of the detection unit and the motors circuits to prevent it from damaging the FPGA. Figure 1.9 demonstrates a SN74LS245n internal Circuit and pins.

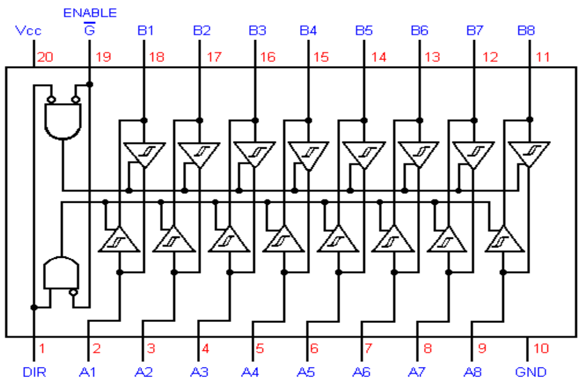


Figure 1.9 SN74LS245n internal Circuit and pins [17]

ULN2803 is a Transistor Array chip useful to drive high power loads. It consists of an eight NPN Darlington drivers [1]. ULN280 is demonstrated in Figure 1.10.

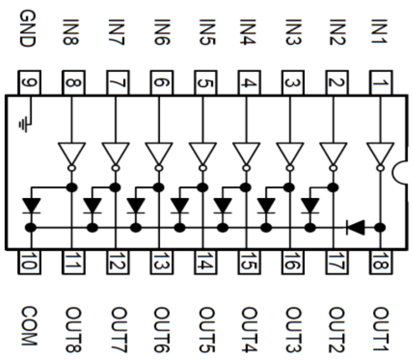


Figure 1.10 : ULN2803 internal Circuit [17]

A Darlington pair is two transistors combined together providing high current gain as 24V. It works as a relay, taking a low level input signals and uses that to provide the current amplification required by the loads, the current that can be drawn at the output is 500 mA [1].

NE555 Timer IC

The latter is an integrated circuit used in numerous applications. The 555 is mainly used either to provide time delay or to generate periodic signals. Figure 1.11 illustrates the pin-out and the internal structure of the 555 Timer IC. [29]

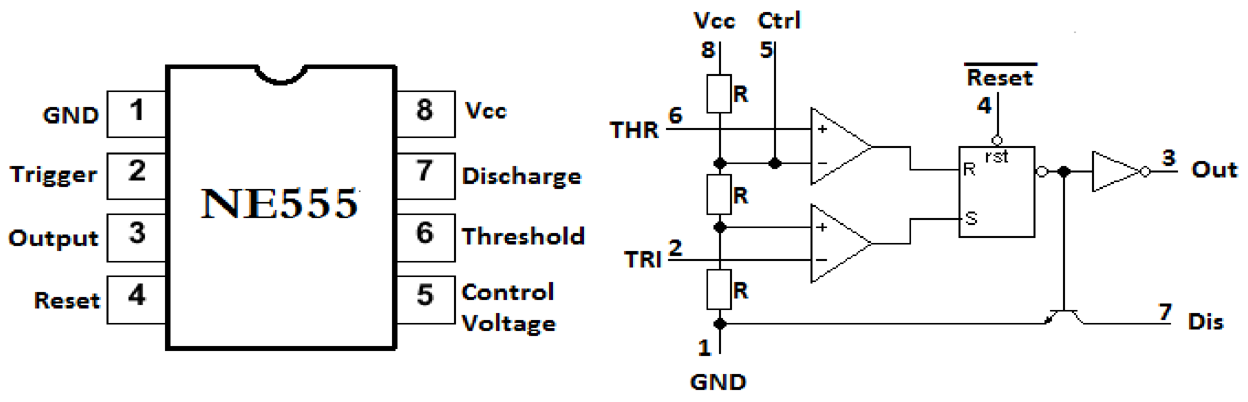


Figure 1.11 NE555 Timer Integrated Circuit [29]

The NE555 has three operating modes: Bi-stable, Monostable and A-stable mode. The two most used are Mono-stable and A-stable modes.

Chapter II

Home-Automation System Design

2.1 Overall system design

The house is handcrafted (using wood) with one chamber with a door, a window; having electrical curtains, and a gas, light and infra-red sensors. The door has two LDRs and IR sensors controlling an electrical curtain. The window is similar to the door; however, there is no IR sensor. This system divided into two parts, the off-chip and the on-chip systems as demonstrated in Figure 2.1.

The off-chip hardware system is implemented on the breadboard using analogue and digital circuits; it includes the bidirectional octal buffers, the sensing units, data acquisition units and motor driving units. The on-chip hardware system is the Qsys system that is implemented onto the DE10 standard board. The on-chip design is based on the Nios II/e

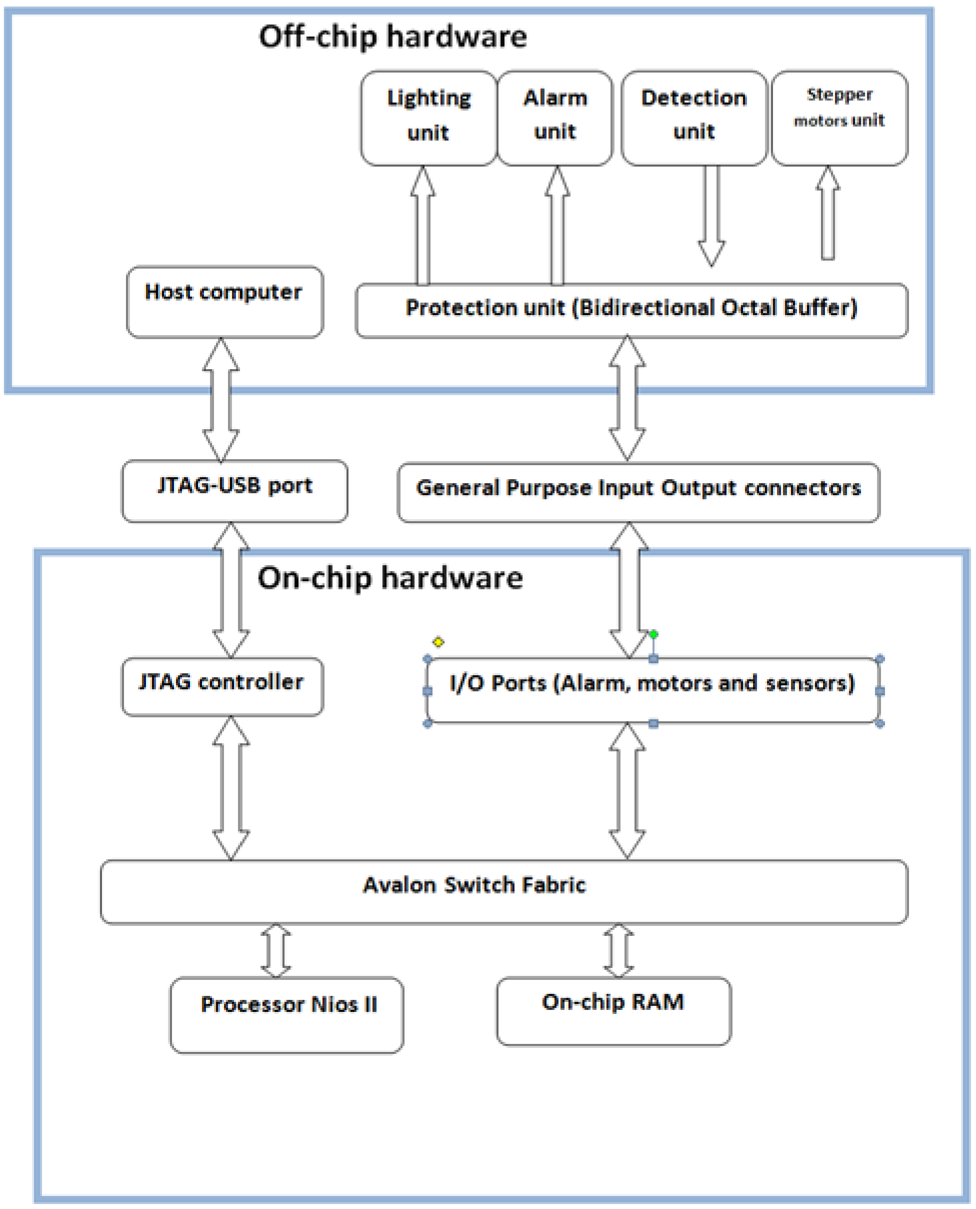


Figure 2.1 global system design diagram

which controls the off-chip hardware through the ribbon cable plugged into the DE10’s GPIO connector as shown in Figure 2.2.

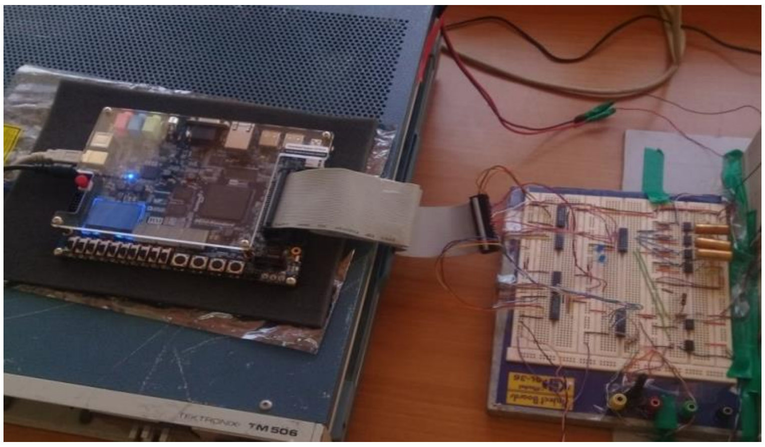


Figure 2.2 The Fpga connected to the off-ship system through a ribbon cable

2.2 Off- chip system

The off-chip circuit is composed of: Detection Unit (sensors), Electric Curtains, Motor Driving Units and a Protection Unit. The breadboard holding the off-chip system is shown in Figure 2.3 below.

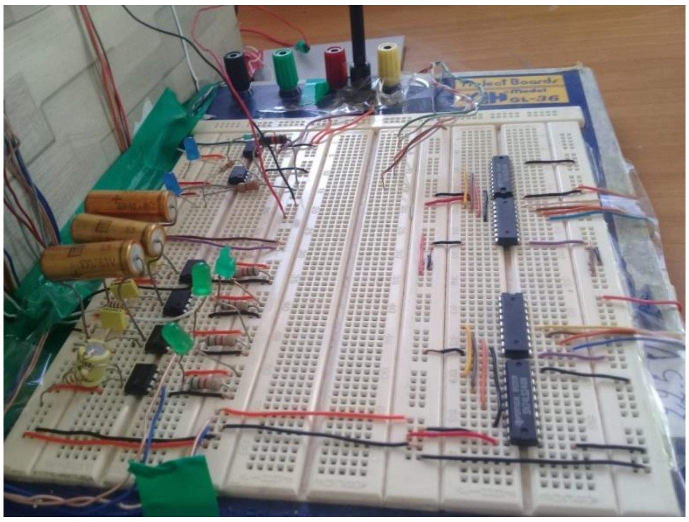


Figure 2.3 The off-ship system

2.2.1 Detection Unit

Four LDR sensors are used to detect the states of whether the door or the window is opened or closed; when the two LDR signals are logic ‘1’ means that the curtain is down and logic ‘0’ means the curtain is open. Figure 2.4 shows the circuit for the LDR sensors.

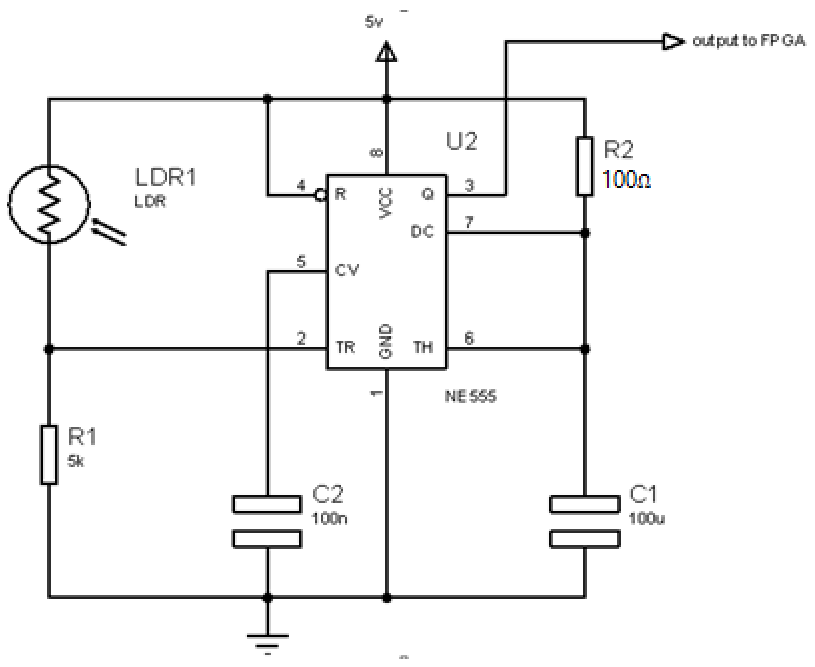


Figure 2.4 The circuits of the LDR

An IR sensor is used to detect if a person stands before the door, system opens it for him. Figure 2.5 shows the circuit for the IR sensors.

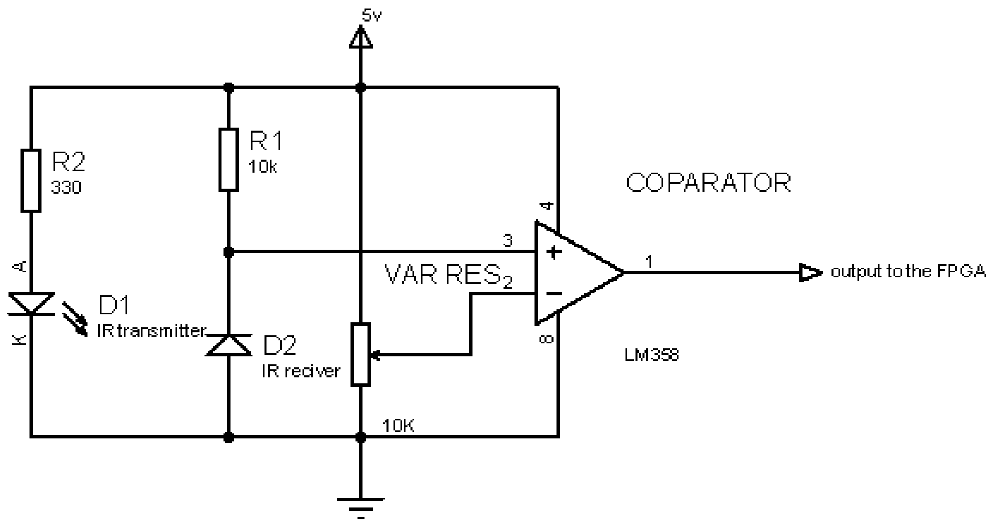


Figure 2.5 The circuits of the IR sensors

A gas sensor is used to detect gas leakage or smoke, then the system automatically give a buzzer and light alerts and opens the window at the same time. The house lighting is controlled automatically, through the same process. Figure 2.6 shows the gas detection circuit

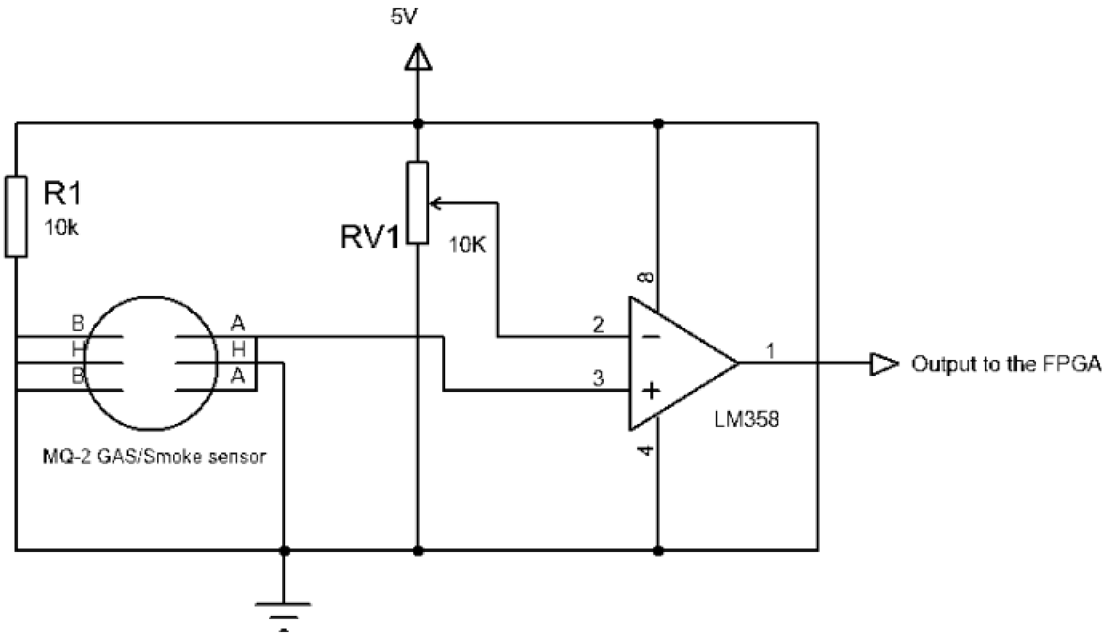


Figure 2.6 Gas detection circuit

2.2.2 Electric Curtains

The Electric Curtain Unit is designed to open or close the door or the window of the house, relying on the detection unit which captures and transmits information to the FPGA. The controller on the FPGA will then send back commands to the house actuators based on the data received from the sensors.

Two 24V dc stepper motors are used In order to raise the electric curtain of the door and the window; the motors require high current to turn, it cannot be directly connected to the DE10 which has low power outputs. Thence, the ULN2803 driver chip is used to power the motors based on the FPGA outputs to drive these motors.

The controller on the FPGA will then send back commands to the house actuators based on the data received from the sensors.

Motor Driving Units

The motor driving unit is used to drive motors of the door and window electrical curtains. It includes amplifiers. The ULN2803 is used to drive the 24V from the 3.3V of the FPGA. Figure 2.7 shows the electric curtain unit circuit.

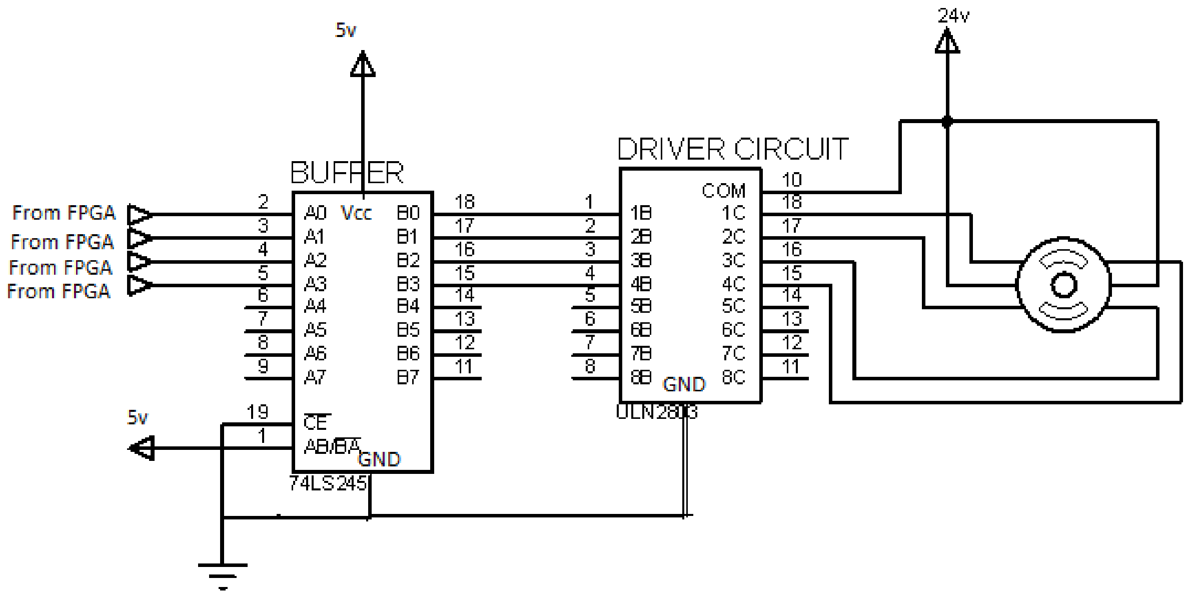


Figure 2.7 the electric curtain unit circuit.

2.2.3 Protection unit

To protect the DE10 board and it’s integrated Equipments; octal Buffers with 3-state outputs (SN74LS245) are used to isolate the input current and the output one for the detection unit and additionally, in the motors circuits**.** Prevent it from damaging the FPGA.

In order to lower/raise the electric curtain of both the door and the window, two 24V dc stepper motors are used. These motors require high current to turn and cannot be connected directly to the FPGA which has low power outputs. The ULN2803 driver chip is used to power the motors based on the FPGA outputs to drive these motors.

2.3 On-ship system (Qsys System)

An embedded system consists of two parts, hardware and software (firmware). Each part is useless with the absence of the other. After building the hardware system, the software takes its roles to manipulate the data and take decision accordingly, it can be written in assembly language or high level language to be executed by the microprocessor.

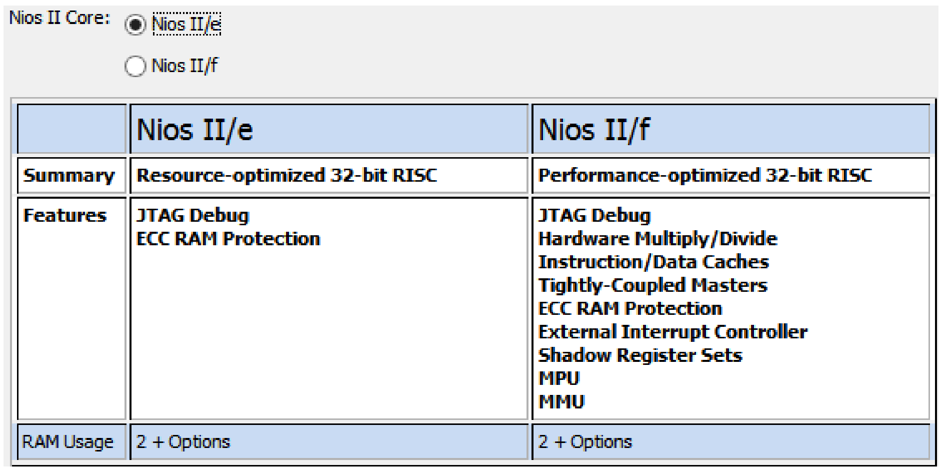
The firmware describes the behaviour of the different components used. It is programmed using Nios II Assembly language, containing multiple routines shared between different tasks. The output of the project firmware depends on the received data from the sensors. It generates an output that controls the house’s actuators through the off-chip system.

The ON-chip of our system is consisted of the Qsys system includes the Nios II/e CPU which is applicable to our system and the on-chip memory and I/O peripherals.

2.3.1 Steps of the design

Quartus prime 16.1 lite edition is a free edition of Quartus Prime from Intel in which we are going to work. It is compatible with the Cyclone V FPGA.

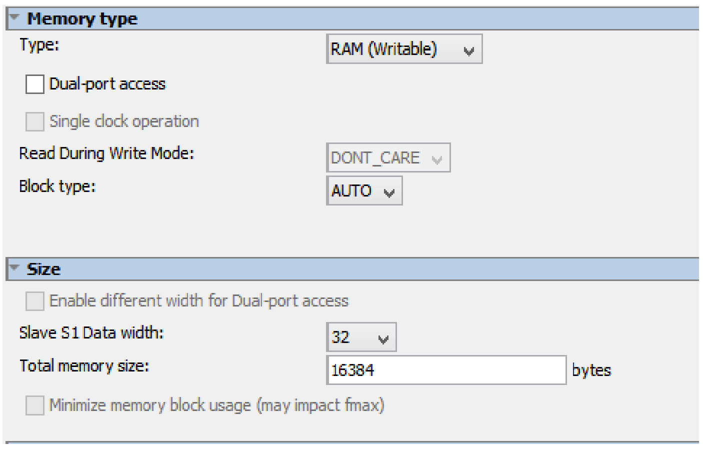
In the Qsys platform designer tool, from component library given we select Nios 2/e processor as demonstrated in Figure 2.8. From the window we select all specifications of the processor; the data width, on chip memory size, instruction type. Nios 2 has a Jtag interface.


Nios II Core: [✓ Nios II/e] Nios II/f

	Nios II/e	Nios II/f
Summary	Resource-optimized 32-bit RISC	Performance-optimized 32-bit RISC
Features	JTAG Debug ECC RAM Protection	JTAG Debug Hardware Multiply/Divide Instruction/Data Caches Tightly-Coupled Masters ECC RAM Protection External Interrupt Controller Shadow Register Sets MPU MMU
RAM Usage	2 + Options	2 + Options

Figure 2.8 Nios 2/e processor from the component list

From the same component list we add memory and PIOs to our design as demonstrated in figures below. The on chip memory added is a 16k byte, and 32-bit data width of type RAM.



Memory type

Type: RAM (Writable)

[ ] Dual-port access

[ ] Single clock operation

Read During Write Mode: DONT_CARE

Block type: AUTO

Size

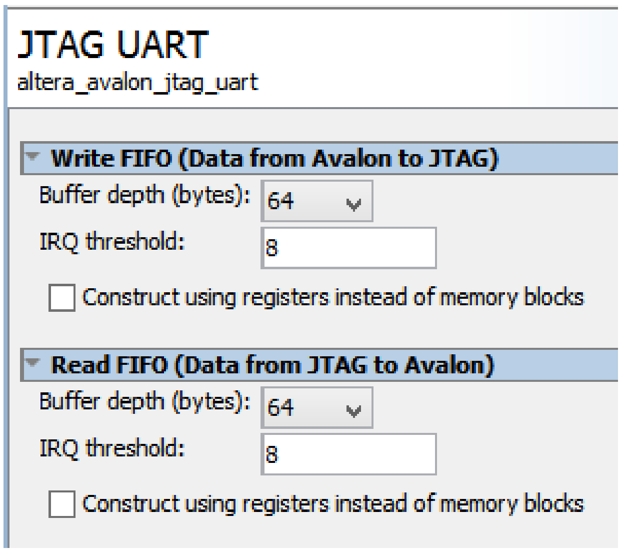
[ ] Enable different width for Dual-port access

Slave S1 Data width: 32

Total memory size: 16384 bytes

[ ] Minimize memory block usage (may impact fmax)

Figure 2.9 adding a Memory (RAM)



JTAG UART

altera_avalon_jtag_uart

Write FIFO (Data from Avalon to JTAG)

Buffer depth (bytes): 64

IRQ threshold: 8

[ ] Construct using registers instead of memory blocks

Read FIFO (Data from JTAG to Avalon)

Buffer depth (bytes): 64

IRQ threshold: 8

[ ] Construct using registers instead of memory blocks

Figure 2.10 Jtag for communication between the host computer and target device.

We then add two PIOs to the design.

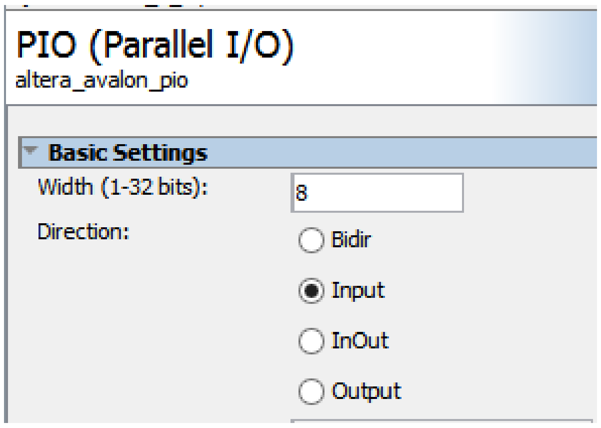

PIO (Parallel I/O)
altera_avalon_pio
Basic Settings
Width (1-32 bits): 8
Direction: ( ) Bidir [✓ Input] ( ) InOut ( ) Output

Figure 2.11 adding an input PIO with 8 bit data width to our system

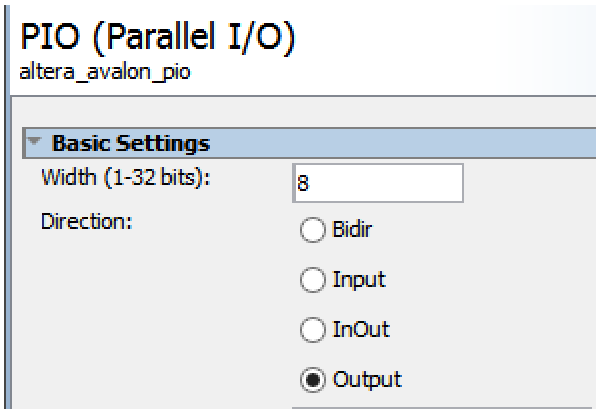

PIO (Parallel I/O)
altera_avalon_pio
Basic Settings
Width (1-32 bits): 8
Direction: ( ) Bidir ( ) Input ( ) InOut [✓ Output]

Figure 2.12 Adding an output PIO with 8 bit data width to our system

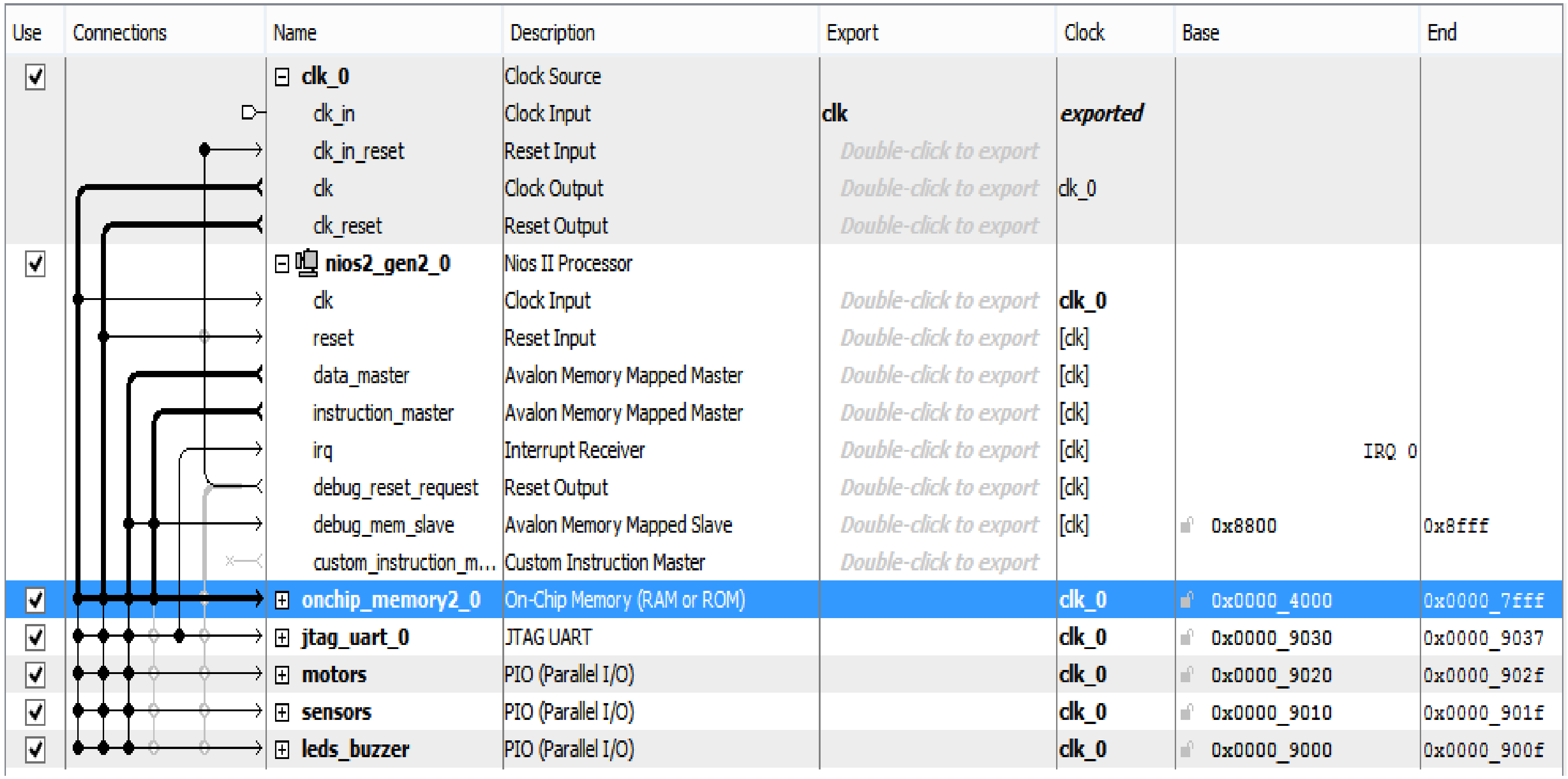
The number of ports and data width of PIOs added is according to our required output and inputs for motors alarms and for sensors. Connections to external peripherals are grouped together in three PIOs as follows:

PIO$_1$: **motors**: contains eights output signals to control the two stepper motors, four signals each one.

PIO$_2$: **sensors**: contains six input signals to read the data from the different sensors used, which are four LDRs, one IR sensor, and one gas sensor.

PIO3: **leds_buzzer**: contains five output signals to control four LEDs and a buzzer.

We connected our processor to a clock and connect all peripherals to the processor. The overall design is demonstrated Figure 2.13 as one Qsys system (SOPC). The Inputs and outputs peripherals are memory-mapped to the addresses demonstrated in the Base/End columns. This system contains no Non-SOPC part.



Use	Connections	Name	Description	Export	Clock	Base	End
✓		⊟ **clk_0**	Clock Source				
		clk_in	Clock Input	**clk**	***exported***		
		clk_in_reset	Reset Input	*Double-click to export*			
		clk	Clock Output	*Double-click to export*	clk_0		
		clk_reset	Reset Output	*Double-click to export*			
✓		⊟ **nios2_gen2_0**	Nios II Processor				
		clk	Clock Input	*Double-click to export*	**clk_0**		
		reset	Reset Input	*Double-click to export*	[clk]		
		data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]		
		instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]		
		irq	Interrupt Receiver	*Double-click to export*	[clk]	IRQ 0	
		debug_reset_request	Reset Output	*Double-click to export*	[clk]		
		debug_mem_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x8800	0x8fff
		custom_instruction_m...	Custom Instruction Master	*Double-click to export*			
✓		⊞ **onchip_memory2_0**	On-Chip Memory (RAM or ROM)		**clk_0**	0x0000_4000	0x0000_7fff
✓		⊞ **jtag_uart_0**	JTAG UART		**clk_0**	0x0000_9030	0x0000_9037
✓		⊞ **motors**	PIO (Parallel I/O)		**clk_0**	0x0000_9020	0x0000_902f
✓		⊞ **sensors**	PIO (Parallel I/O)		**clk_0**	0x0000_9010	0x0000_901f
✓		⊞ **leds_buzzer**	PIO (Parallel I/O)		**clk_0**	0x0000_9000	0x0000_900f

Figure 2.13 Nios II system design in Qsys

Avalon Switch Fabric

It is used as the interface to its embedded peripherals similar to a traditional bus in a processor-based system, which is only one bus master access the bus at a time.

After that, saving and generating the design is by clicking on ‘generate’ from the list and choosing the VHDL synthesis file type.

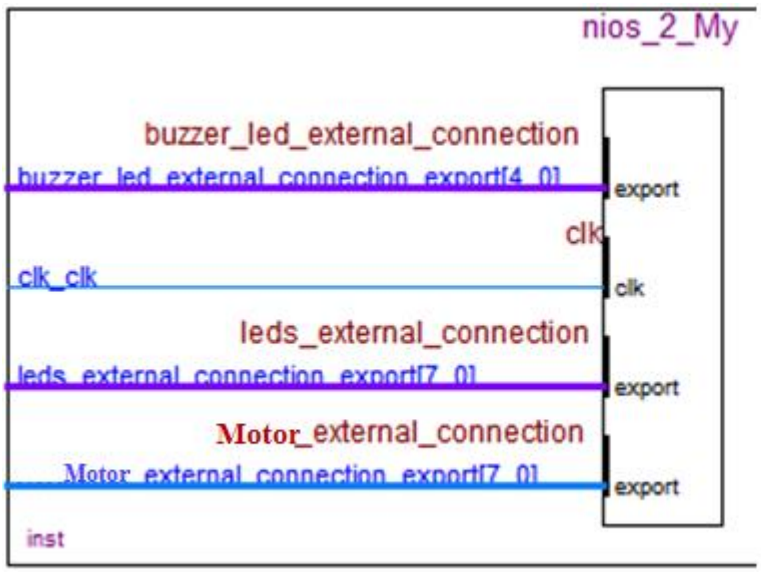
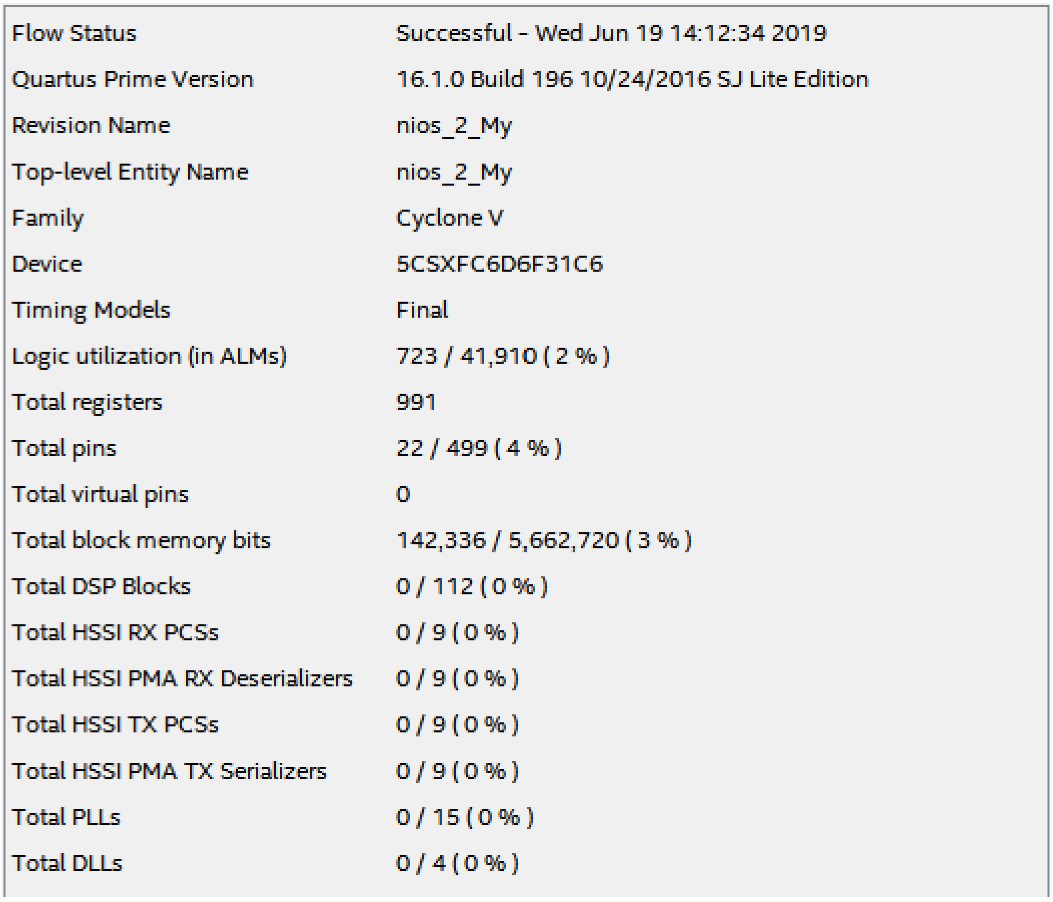

nios_2_My
buzzer_led_external_connection
buzzer_led_external_connection_export[4..0]
export
clk
clk_clk
clk
leds_external_connection
leds_external_connection_export[7..0]
export
Motor_external_connection
Motor_external_connection_export[7..0]
export
inst

Figure 2.14 Nios II block diagram from Qsys

Next task is to generate our design to obtain a block diagram from Qsys representing our Nios II system. Figure 2.14 demonstrates the block diagram generated which we can use in Quartus prime block diagram builder.

Back in the project on Quartus prime, we open new “schematic and diagram” file. Browsing components and blocks we select our block. Next we add input and output pins and save and compile. Next, in the pins planer we assign our inputs and outputs to the GPIO and compile.



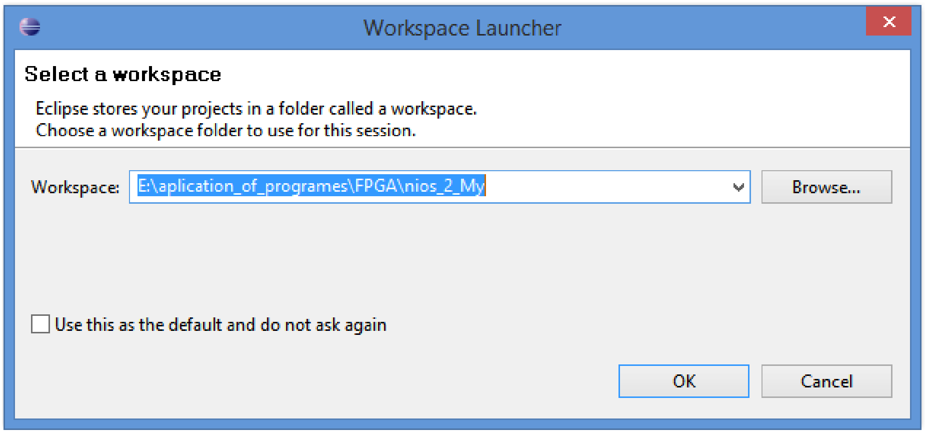
Flow Status	Successful - Wed Jun 19 14:12:34 2019
Quartus Prime Version	16.1.0 Build 196 10/24/2016 SJ Lite Edition
Revision Name	nios_2_My
Top-level Entity Name	nios_2_My
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	723 / 41,910 (2 %)
Total registers	991
Total pins	22 / 499 (4 %)
Total virtual pins	0
Total block memory bits	142,336 / 5,662,720 (3 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 2.15 compilation summary of the designed Nios system

The summary in Figure 2.15 demonstrates that the design has consumed 4% of total pins of the FPGA, 3% of memory blocks and 2% ALMs to build our design.

2.3.2 Software implementation

NIOS II Software Build Tools for Eclipse is used to create a C program for our Nios 2. Eclipse compiler compiles the C code to Nios 2 assembly language. Inputs and outputs are memory mapped. Figure 2.16 shows the dialog box to open eclipse software.


Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: E:\aplication_of_programes\FPGA\nios_2_My

Figure 2.16 opening the designed project to write the program code

After initialisation, the processor continuously reads the status of the input sensors, and according to the read data, it will execute the specific task. The tasks include opening/closing the door, alarms, and lights… as described below.

The program running on the Nios 2 to control the home is described by means of the flowchart in Figure 2.17.

The flowchart below describes the flow of the control system. The CPU read the sensor signals, if gas sensor is triggered it activates the alarm routine (buzzer and yellow LED's flashes) and if the window is closed it gives sequence to the window stepper motor to open it, permitting the gas out. After that, it checks again, and if there is no gas it shut down alarms and close back the window. If the IR sensor is triggered, it opens the door and turn on the blue LED's, otherwise, it closes the door and turn off the blue LED.

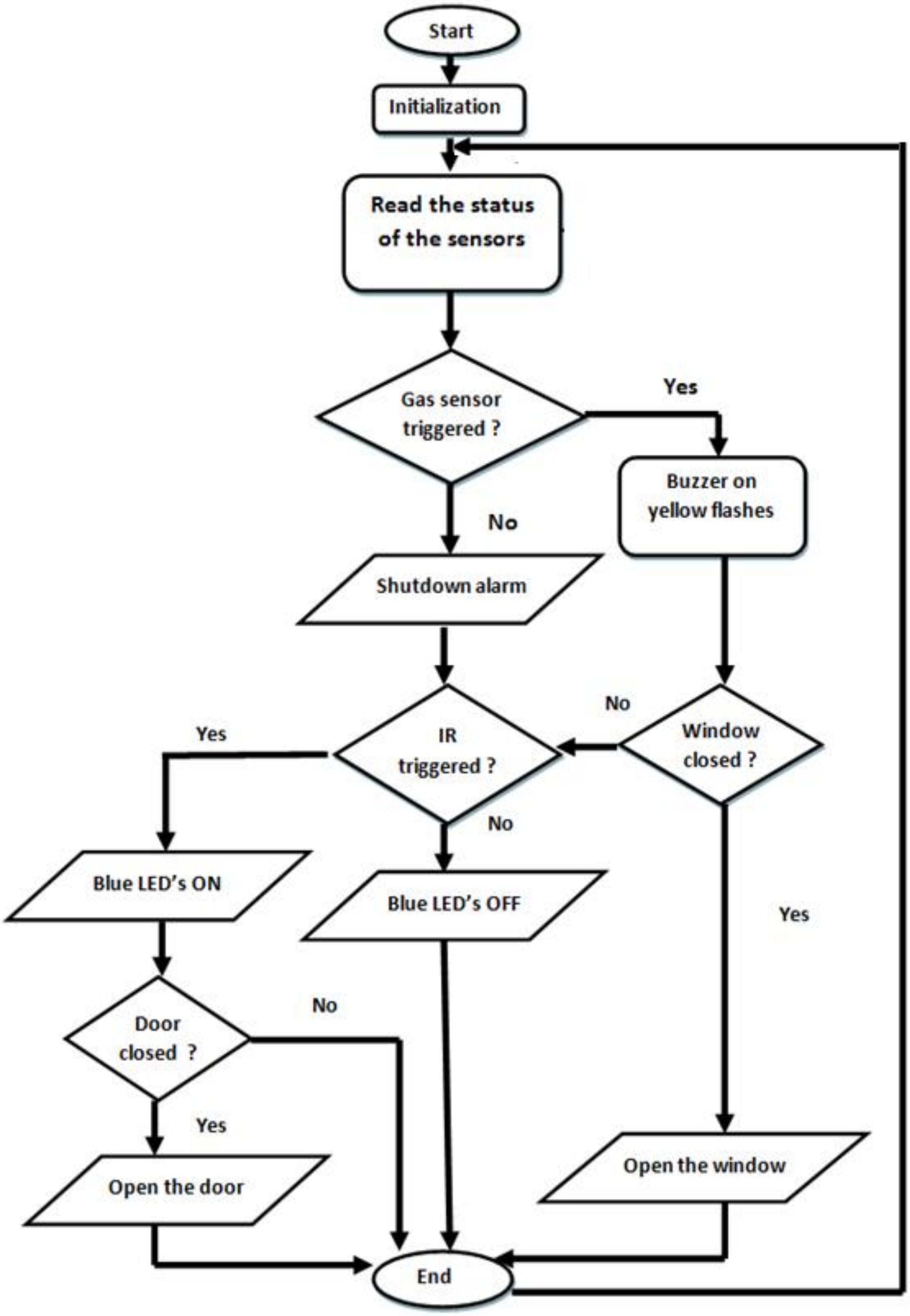


Figure 2.17 workflow of the main program

Figure 2.18 below demonstrates the home prototype.

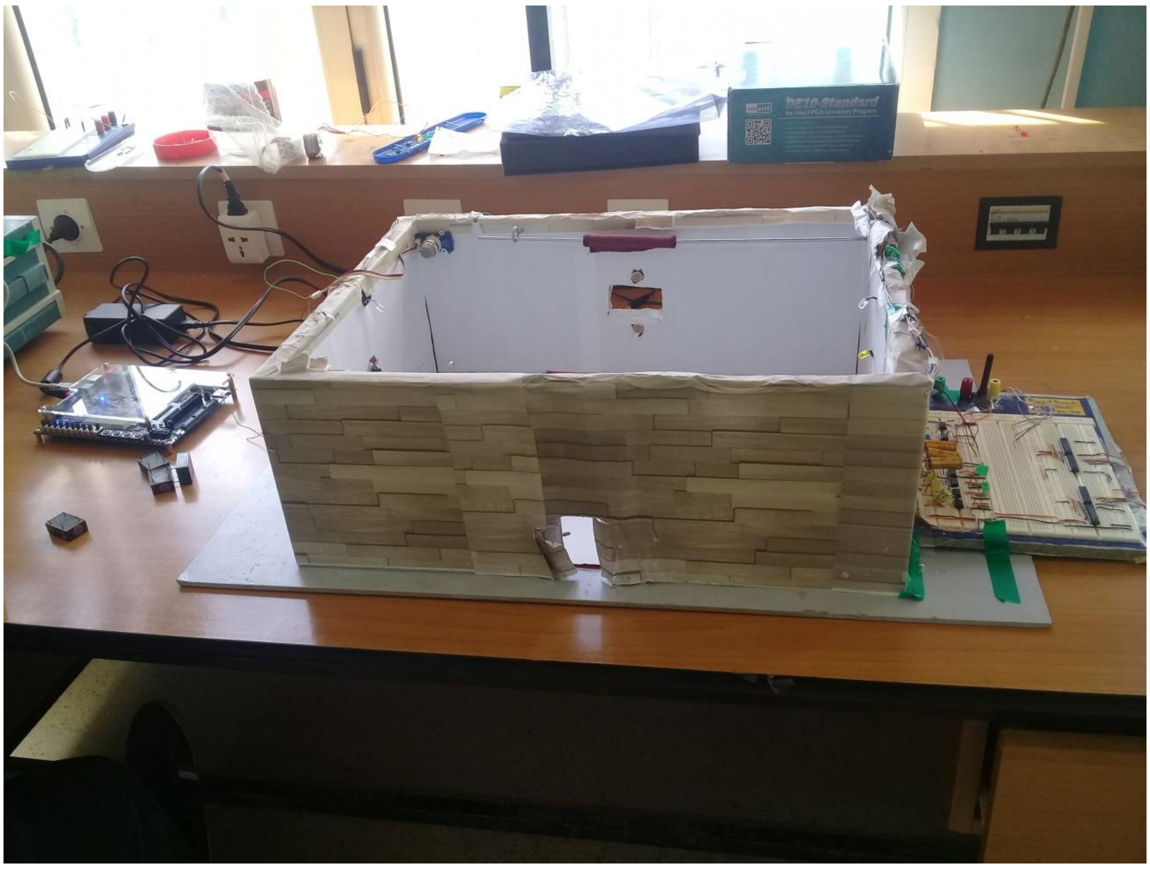


Figure 2.18 overall system design

Chapter III

OpenCV Based Fire Detection

3.1 Overview

In this chapter, we will design a fire detection system using embedded-Linux on the DE10 board. Application of fire detection as tool has increased. These detection methods which are based on electronic sensors are usually depending on heat and pressure sensors. However those methods have fatal errors where they will only work when a certain condition has been reached. The worst case scenario is when the sensors are damaged or not being configured properly can cause heavy casualty in case of real fire [18].

Automatic fire detection systems are developed to detect and extinguish fires without human intervention. From among there comes the Computer vision system which is based on image processing.

The Figure 3.1 demonstrates the design we used and its different parts.

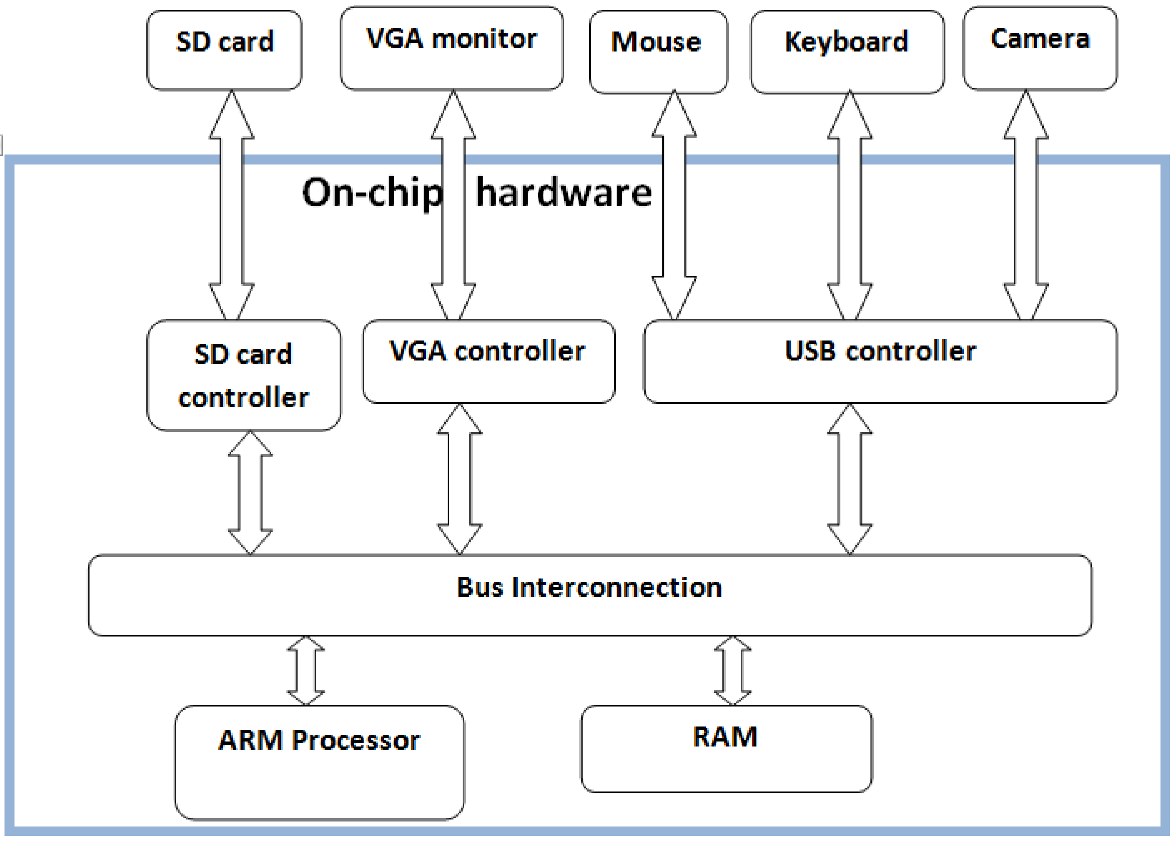
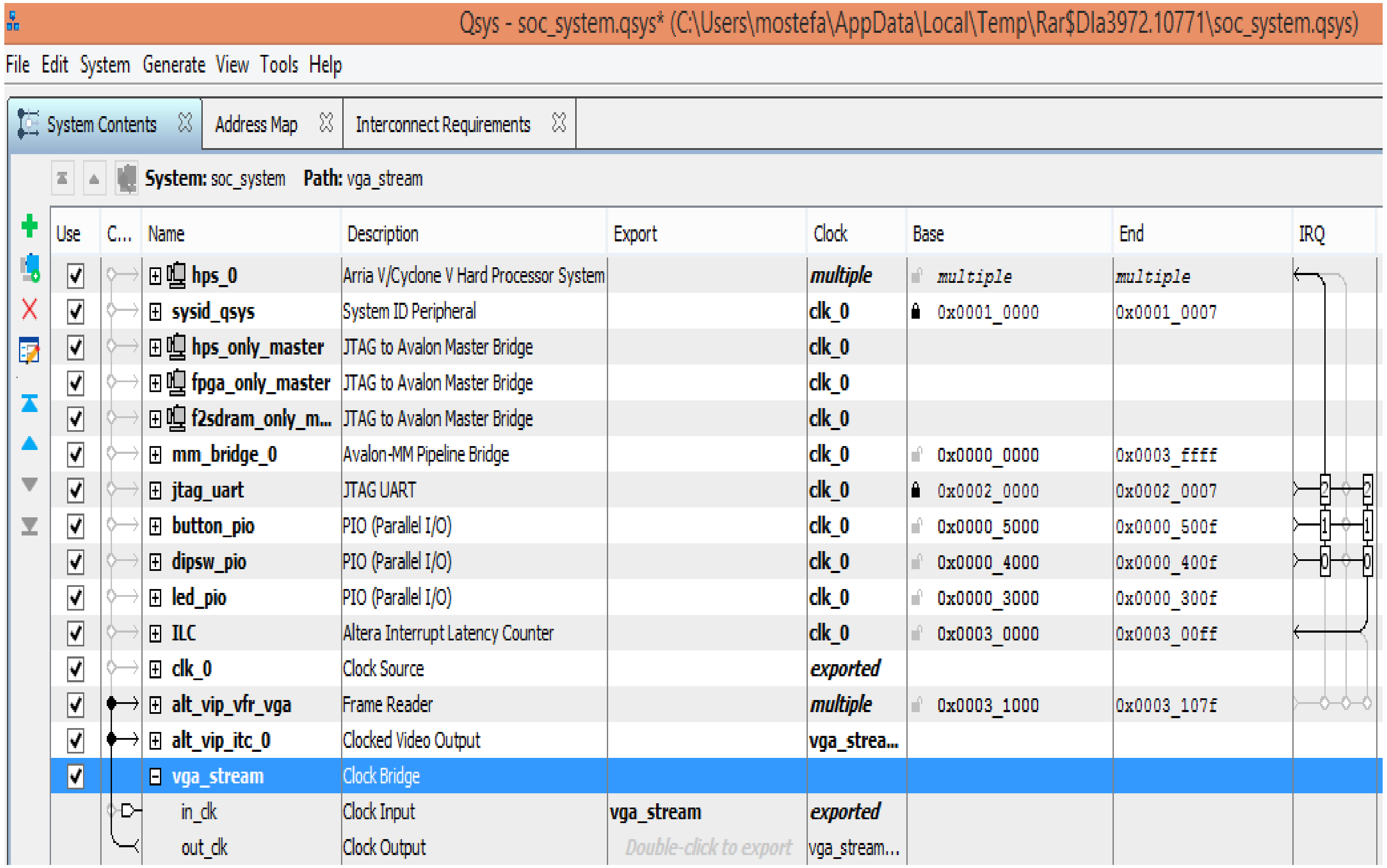


Figure 3.1 Fire detection system design using predesigned HPS

The system contains off-chip hardware (camera, VGA monitor, keyboard, mouse, and an SD-card) in addition to an On-chip system. The on-chip system is the HPS system already exist on the DE10 board with the ARM processor and DDR3 RAM.

3.2 OpenCV based fire detection system

To work with OpenCV we can use LXDE Desktop Image installed in SD Card and run it on HPS (Hard Processor System) included into FPGA. The library of OpenCV has been installed into LXDE Desktop Image. Hence, we can build a code of fire detection system and compile it using command lines. Figure 3.2 below demonstrates the HPS components coming by default with the DE10 board.


Qsys - soc_system.qsys* (C:\Users\mostefa\AppData\Local\Temp\Rar$DIa3972.10771\soc_system.qsys)

File Edit System Generate View Tools Help

System Contents | Address Map | Interconnect Requirements

System: soc_system **Path:** vga_stream

Use	C...	Name	Description	Export	Clock	Base	End	IRQ
✓		hps_0	Arria V/Cyclone V Hard Processor System		*multiple*	*multiple*	*multiple*	
✓		sysid_qsys	System ID Peripheral		clk_0	0x0001_0000	0x0001_0007	
✓		hps_only_master	JTAG to Avalon Master Bridge		clk_0			
✓		fpga_only_master	JTAG to Avalon Master Bridge		clk_0			
✓		f2sdram_only_m...	JTAG to Avalon Master Bridge		clk_0			
✓		mm_bridge_0	Avalon-MM Pipeline Bridge		clk_0	0x0000_0000	0x0003_ffff	
✓		jtag_uart	JTAG UART		clk_0	0x0002_0000	0x0002_0007	2 2
✓		button_pio	PIO (Parallel I/O)		clk_0	0x0000_5000	0x0000_500f	1 1
✓		dipsw_pio	PIO (Parallel I/O)		clk_0	0x0000_4000	0x0000_400f	0 0
✓		led_pio	PIO (Parallel I/O)		clk_0	0x0000_3000	0x0000_300f	
✓		ILC	Altera Interrupt Latency Counter		clk_0	0x0003_0000	0x0003_00ff	
✓		clk_0	Clock Source		*exported*			
✓		alt_vip_vfr_vga	Frame Reader		*multiple*	0x0003_1000	0x0003_107f	
✓		alt_vip_itc_0	Clocked Video Output		vga_strea...			
✓		vga_stream	Clock Bridge					
		in_clk	Clock Input	vga_stream	*exported*			
		out_clk	Clock Output	*Double-click to export*	vga_stream...			

Figure 3.2 HPS system component

3.2.1 OpenCV

The DE10-Standadrd LXDE Desktop BPS has built-in OpenCV library, therefore user can use the library to perform computer vision functions. In addition, the BPS includes the required toolchain for building OpenCV application; therefore developers can directly develop and execute their project on the LXDE Desktop. No cross-compile is required [5].

3.2.2 Ubuntu LXDE Desktop Image

LXDE is a lightweight desktop alternative to Unity, GNOME and KDE. It is ideal for old computers or anyone looking for a fast, lightweight system. LXDE contains the basic features for a stripped-down, yet approachable, desktop environment [19].

3.2.3 Implementation of the system:

Figure 3.3 below demonstrates the system hardware.

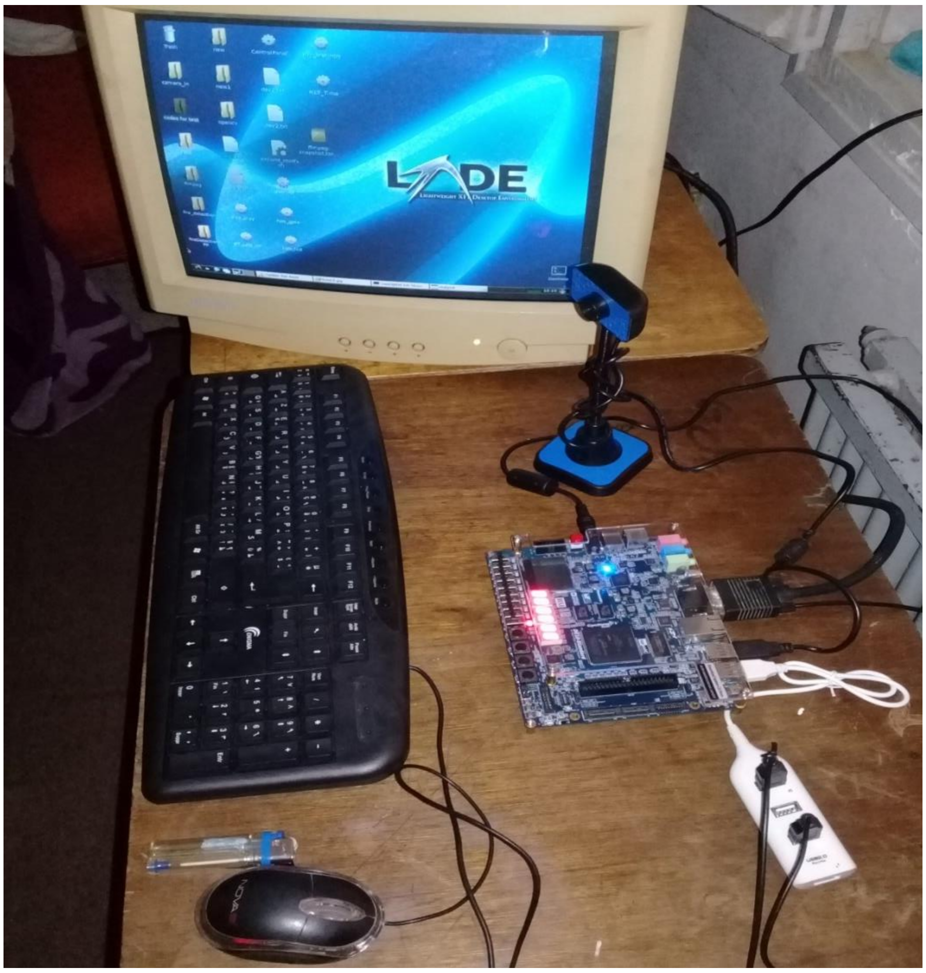


Figure 3.3 Fire detection system

Linux on SD card

After downloading the LXDE Desktop Image from the Terasic website, it is then copied into a microSD card using **Win32 Disk Imager** utility.

Connecting the camera and other peripherals

Now, we can connect the camera and other accessories such as mouse, VGA monitor and a keyboard to the FPGA and then run the system, the Ubuntu LXDE has a set of drivers to read those basic devices.

Fire detection software

Before writing the code, some knowledge must be known about the fire to determine it in each pixel Based on fire characteristics. Accordingly, some researchers studied these data and summarized in some equations to use in building a code of fire detection**.**

Fire is the rapid oxidation of a material in the exothermic chemical process of combustion, releasing heat, light, and various reaction products.

Our fire detection system is implemented using OpenCV based on embedded-Linux. Research demonstrated that flames have specific colour properties. These properties are then extracted from a live video in each frame and used to detect the flames. In this project, RGB and Y, Cb and Cr colour spaces are both used to process live stream images.

In order to create the colour model for fire, several images having fire have been analysed. Since the colour of fire is generally close to red and has high illumination, then we can use this property to derive the required colour model. The program is described with the flowchart bellow in Figure 3.4.

The camera reads a frame and sends it to the CPU; the program treats the frame as three separated basic colours Red, Blue and Green. It converts them to Y, Cb, and Cr model. Using conditions:

$$\mathrm{Y} = 16 + \mathrm{R} * 65.481 + \mathrm{G} * 128.553 + \mathrm{B} * 24.99 \qquad 3$$

$$\mathrm{Cb} = 128 + \mathrm{R} * -37.797 - \mathrm{G} * 74.203 + \mathrm{B} * 112.0 \qquad 4$$

$$\mathrm{Cr} = 128 + \mathrm{R} * 112.00 + \mathrm{G} * -93.7864 + \mathrm{B} * -18.214 \qquad 5$$

Then we can summarize the conditions to four rules:

Rule1: $\left((\mathrm{R}(\mathrm{x,y}) > G(\mathrm{x,y}))\&\&(\mathrm{G}(\mathrm{x,y}) > B(\mathrm{x,y}))\right)$ 7

Rule2: $(\mathrm{R}(\mathrm{x,y}) > 190)\ \&\&\ (G(x,y) > 100)\ \&\&\ (B(x,y) < 140)$ 8

Rule3: $\mathrm{Y}(\mathrm{x,y}) >= Cb(x,y)$ 9

Rule4: $(\mathrm{Cr}(\mathrm{x,y}) >= Cb(x,y)$ 10

If a pixel satisfies the fire detection rules, its output is white, if not it is black pixel. After reaching the last pixel if there is more than 500 pixel of fire it gives the written alarm “fire detected”, otherwise, it goes to the next frame and so on.

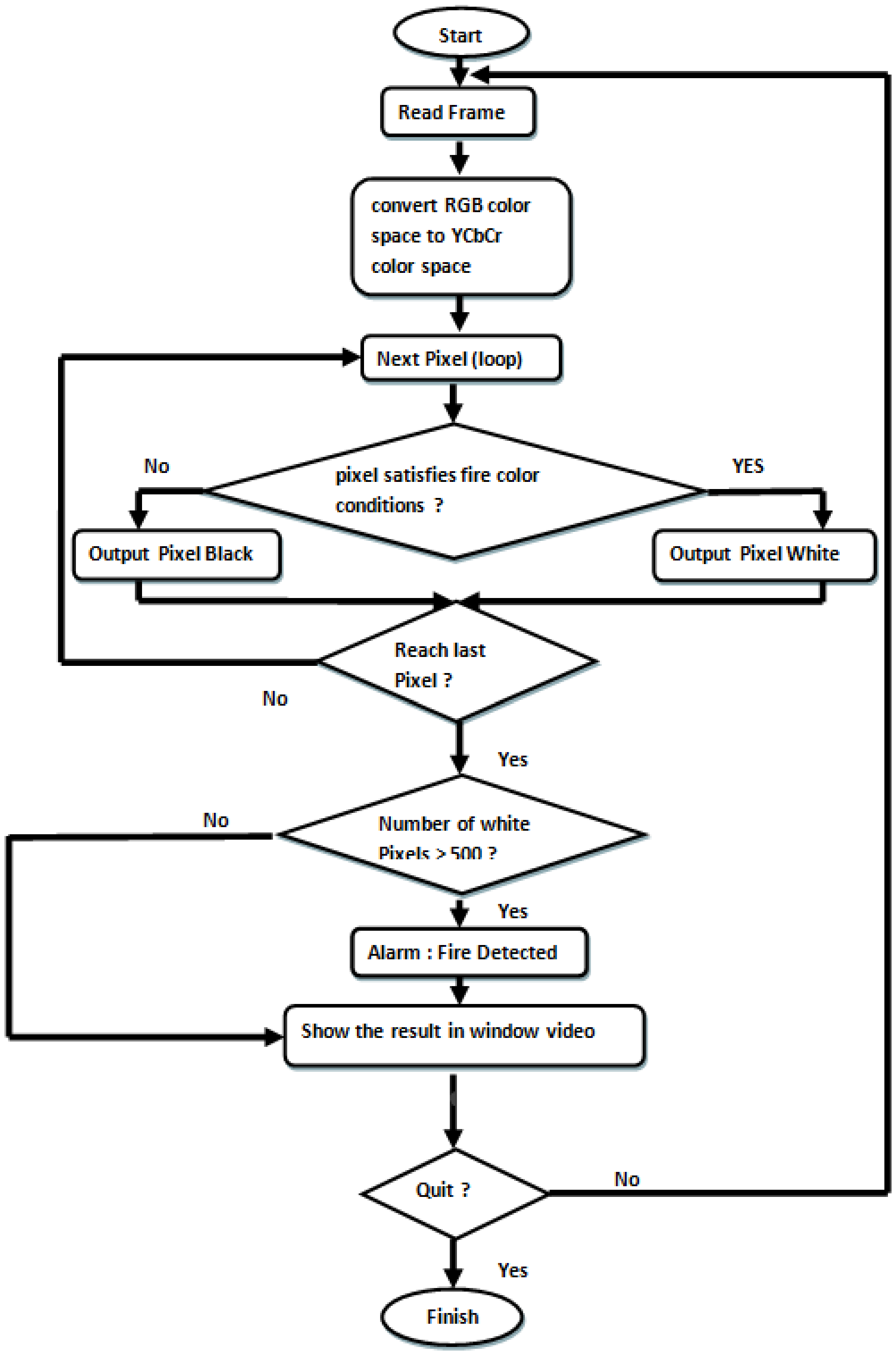
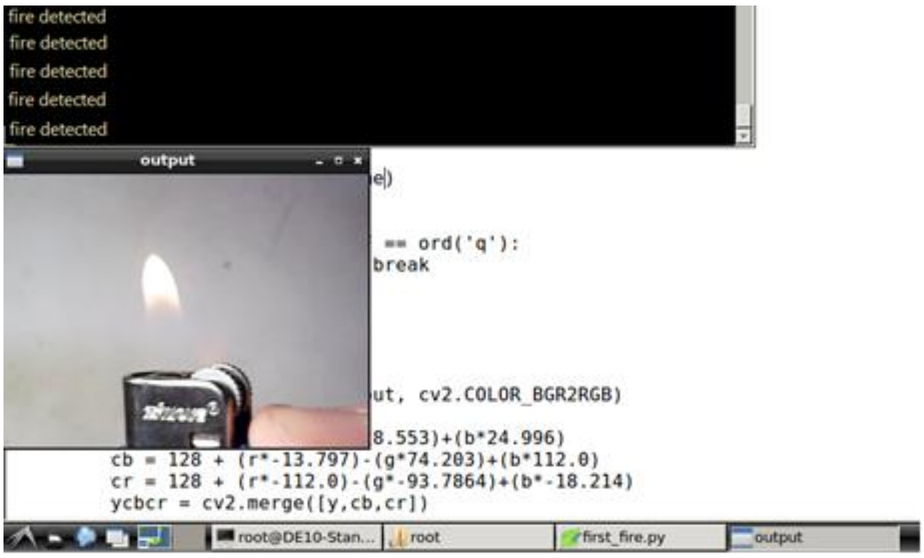

Start
Read Frame
convert RGB color space to YCbCr color space
Next Pixel (loop)
pixel satisfies fire color conditions ?
No
YES
Output Pixel Black
Output Pixel White
Reach last Pixel ?
No
Yes
Number of white Pixels > 500 ?
No
Yes
Alarm : Fire Detected
Show the result in window video
Quit ?
No
Yes
Finish

Figure 3.4 fire detection system flowchart

The execution of the program is shown in the figures bellow. In Figure 3.5 the real image of fire, and in Figure 3.5 it is shown the masked image with only fire is in white and a message says “fire detected”.

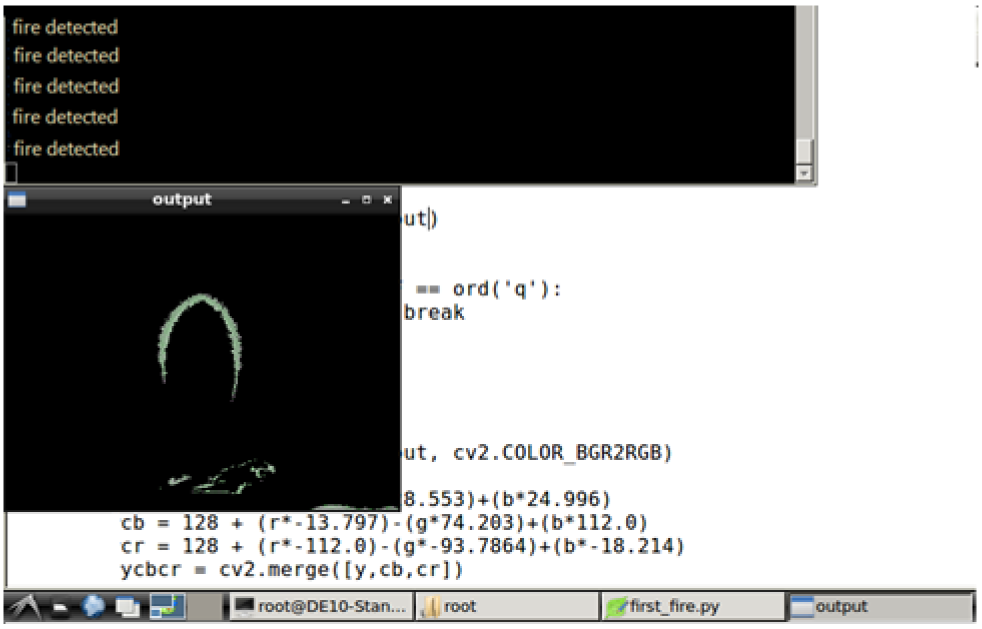


```
fire detected
fire detected
fire detected
fire detected
fire detected
```

```
e|)
 == ord('q'):
break
ut, cv2.COLOR_BGR2RGB)
8.553)+(b*24.996)
cb = 128 + (r*-13.797)-(g*74.203)+(b*112.0)
cr = 128 + (r*-112.0)-(g*-93.7864)+(b*-18.214)
ycbcr = cv2.merge([y,cb,cr])
```

Figure 3.5 fire image with real colours

Here is an image after the mask is on. It shows only fire perimeter in white colour and writing a message “fire detected” as shown in Figure 3.6.



```
fire detected
fire detected
fire detected
fire detected
fire detected
```

```
ut|)
 == ord('q'):
break
ut, cv2.COLOR_BGR2RGB)
8.553)+(b*24.996)
cb = 128 + (r*-13.797)-(g*74.203)+(b*112.0)
cr = 128 + (r*-112.0)-(g*-93.7864)+(b*-18.214)
ycbcr = cv2.merge([y,cb,cr])
```

Figure 3.6 Image from the program execution (masked)

Chapter IV

Conclusion

Our project was a combination of software and hardware design tasks. We have enriched our skills in hardware -by means of implementing different circuits- and software with the Fire detection system. During the process of implementing this system, we have faced a large number of software and hardware problems. Overcoming these problems provided us with a valuable experience and improved our problem solving skills.

The implementation has successfully worked. The response of home system was automatic and at real-time. The detection unit can detect gas leakage, and automatically alerts the owner; then the system automatically generates a buzzer and light alarms and opens the window at the same time. The window curtain opens whenever there is gas leak detected to evacuate the house, and close automatically when there is no more gas. The door curtain was very accurate because of the LDR's. IR sensor works with curtains; therefore if a person passes the IR sensor sends logic '0' to the controller on the FPGA, which will then send back commands as a sequence elevate the door curtain. After he passes the curtain will automatically close.

For OpenCV section, the Fire Detection responds at real-time. The system detects any fire captured by means of the USB camera at any part of the frame. It donates a masked output image black and white for fire, and outputs a text alarm "Fire Detected". The system has perfectly given the expected results.

For the future, the home automation and fire detection system can be integrated in one system running on the embedded-Linux within the DE10 board. Knowing the fact that Linux is running on HPS; this can be done by configuring HPS using Qsys tool and generating custom component for I/O peripherals, sensors and actuators on FPGA to connect HPS with FPGA. Also, creating drivers for these components on Linux kernel is a must, to permit Linux recognize them, as the DE10 board supports Ethernet access, an access point can be added to the system and acts as an intermediary device between the user Smartphone and the FPGA. A streamed video to the user smartphone can be developed using IP camera, the user can watch a live video of the house. An Android application can be developed to permit the user to control all the home lights and curtains and Camera from distance with an easy reliable interface. As an extension for the OpenCV system, a face recognition algorithm can be used. A camera at the door captures and processes the face of the person standing in front of the door, the control system opens the door if the person is authorised or deny access otherwise. Several cameras can be put inside the house and outside to accomplish different tasks like face recognition, fire detection, surveillance or live streaming.

References

[1] R. Melek BOULAHIA et F. KIBECHE, *uClinux-based Embedded System for Home Automation*, Boumerdes, 2017.

[2] H. Stark, Is It Safe To Be Smart? The Security In Home Automation, 06 december 2017, access date: 2019 Online: https://www.huffpost.com/entry/is-it-safe-to-be-smart-th_b_11234416.

[3] R. Wisniewski, Synthesis of compositional microprogram control units for programmable devices, Zielona Góra, 2009.

[4] Terasic, terasic, 2018, access date: may 2019, Online: http://www.terasic.com.

[5] Intel, "Terasic Cyclone V SoC Development Kit with HSMC Connector (DE10-Standard)," 16 October 2018. Online: www.intel.com.

[6] S.-A. Andersson, EE Times, 1 august 2013, access date: june 2019, Online: https://www.eetimes.com/document.asp?doc_id=1280290#.

[7] Intel, AN 796: Cyclone V and Arria V SoC Device Design Guidelines, 20 february 2017. access date: june 2019, Online: https://www.intel.com/content/www/us/en/programmable/documentation/doq1481305867183.html#sxr1481303255441.

[8] G. Bradski et A. Kaehler, Learning OpenCV, M. Loukides, Éd., Sebastopol, California: O'Reilly Media, 2008.

[9] H. Johansson, Evaluating Vivado High-Level Synthesis on OpenCV Functions for the Zynq-7000 FPGA, Malardalen , 2015.

[10] S. Tiwari, Flame Detection using Image Processing Techniques, , Laxmangarh.

[11] Linux operating system, searchenterprisedesktop, December 2016. access date: 2019, Online: http://searchenterpriselinux.techtarget.com/definition/Linux.

[12] The Linux Foundation, what is Linux, 2016, access date: 18 June 2019, Online: https://www.linux.com/what-is-linux.

[13] G. Winn, Bluetooth, Echolo, access date: June 2019, Online: https://echolo.io/category/bluetooth/.

[14] V. Jain, Learn the Working of a Gas Sensor, 15 Mars 2013. access date: 4 Mai 2019, Online: https://www.engineersgarage.com/insight/how-gas-sensor-works.

[15] Electrical4U, Electrical4u, 2019, access date: 18 June 2019, Online:

https://www.electrical4u.com/light-dependent-resistor-ldr-working-principle-of-ldr/.

[16] ELECTRICAL TECHNOLOGY, What is a Stepper Motor? Types, Construction, Operation & Applications, 2016, access date: June 2019, Online: www.electricaltechnology.org.

[17] National Instruments, fpga-fundamentals, 19 Mar 2019, access date: june 2019, Online: http://www.ni.com/en-lb/innovations/white-papers/08/fpga-fundamentals.html.

[18] K. Poobalan et S.-C. Liew, FIRE DETECTION ALGORITHM USING IMAGE PROCESSING TECHNIQUES, International Conference on Artificial Intelligence and Computer Science ,2015.

[19] C. Hoffman, how to install the lightweight lxde desktop on ubuntu, "How-to geek", 2017. access date: June 2019, Online: https://www.howtogeek.com/107368/how-to-install-the-lightweight-lxde-desktop-on-ubuntu/.

[20] L. Schaffer, Z. Kincses et S. Pletl, FPGA-based Low-Cost Real-Time, vol. 1, n° %140, p. 4, 2017.

[21] L. Phifer et J. Trulove, IEEE 802 Wireless Standards, February 2006.

[22] B. L. arbi et T. C. Nassim, *FPGA-based Real-Time Video Processing Framework,* Boumerdes, 2016.

[23] T. Walid and A. Walid, *Remote Control of an SoPC-Based Home Automation System,* Boumerdes, 2017.

[24] P. P. Chu, in Embedded SoPC design with Nios II processor and VHDL examples, A JOHN WILEY & SONS, INC, 2011.

[25] K. Dawson-Howe, A PRACTICAL INTRODUCTION TO COMPUTER VISION WITH OPENCV, Chichester, Southern Gate: John Wiley & Sons Ltd, 2014.

[26] Internet archive, Internet archive wayback machine, 2004, access date: Jun 2019, Online: https://web.archive.org/web/20070412183416/http://filebox.vt.edu/users/tmagin/history.htm.

[27] J. W. C. K. Y. Lee, Remote-Controlled Home Automation System, chez *SICE Annual Conference in Fukui,* Fukui, Japan, 2003.

[28] Intel FPGA, From Qsys to Quartus, Intel FPGA, access date: June 2019, Online: https://www.youtube.com/watch?v=35vDSIS-LOI.

[29] NXP Semiconductors, NE555 Datasheet, 1994.

[30] Semantic Scholar, Implementation of the CUSUM Algorithm on FPGA for Transient Signal Detection, 2012, access date: July 2019, Online: https://www.semanticscholar.org/paper/Implementation-of-the-CUSUM-Algorithm-on-FPGA-for-Li/75e4e53e13742a1d8f320f4c2f833351bd90cdca.