Ping pong ball catcher

Anders Jensen[1], Jorge Rodriguez Marin[2], and Kim Lindberg Schwaner[3]

University of Southern Denmark
Faculty of Engineering
EMB1

October 27, 2014

[1]anje111@student.sdu.dk
[2]jorod14@student.sdu.dk
[3]kschw10@student.sdu.dk

Abstract

In the following pages the design and implementation process of a Lego brick sorter is described. The project consists in being able to recognize a Lego brick's color and use an actuator so the Lego brick will be collocated in a defined position. For this purpose, a FPGA will be programed to read the color from a photodiode and move a servo. The requirement of make the project modular, facing other future projects, a main PCB board is designed so extension boards can be used with the FPGA. The main PCB power the extension modules at the same time it is used for the communication between them and the FPGA.

Contents

Chapter 1

Introduction

1.1 Overall description

The goal of this project is catching a ping pong ball released above a platform, on which three piezoelectric elements are placed in a triangular pattern as shown in Figure 1.1. Once the ball has hit the platform once, each of the piezo elements will generate a signal and permit triangulation of the impact point. When the impact point has been found, a net will catch the ball when it returns toward the platform after the first bounce.

Overall the different tasks of the project are thus:

- Calculating the xy-coordinate of the impact point
- Calculating the hight the ball will bounce back to
- Moving an arm – holding a net – to catch the ball after it has bounced once

1.1.1 Report structure

The report is organized so it starts with the explanation of the project's methodology and the equipment used (2), followed by the description of the design and implementation of the Main PCB (**??**) and the Control-Brick-Sorter Module PCB (**??**) along with the FPGA's programming work (**??**) and the PC program (6) developed and used as an interface between the FPGA to finish with the conclusions and the goals achieved (7).

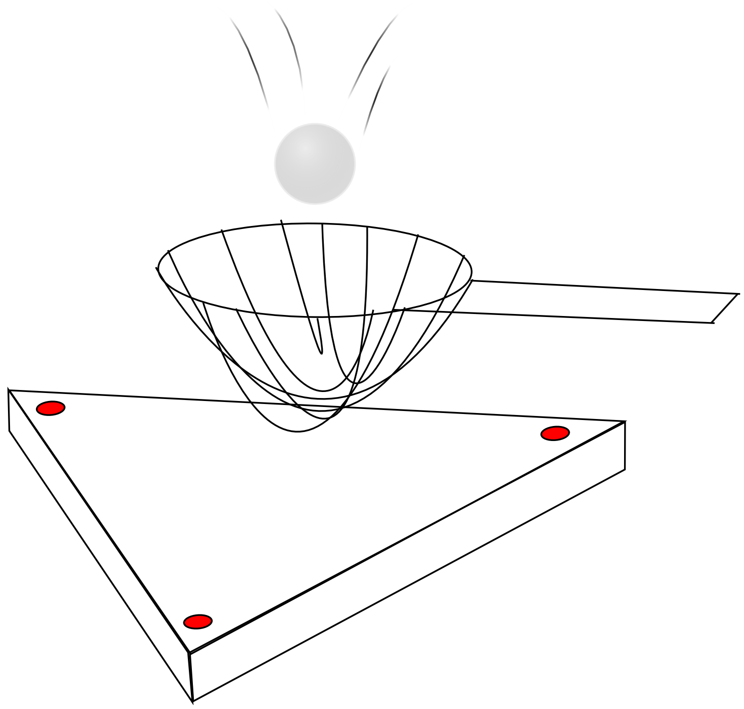

Figure 1.1: Side view of the platform surface area and the net. The red markings are piezoelectric sensors.

Chapter 2

Methodology and Equipment

In order to complete the project a number of methods and materials were used.

2.1 Methodology

The whole project was divided in two *subprojects*: the main board **??** and the brick sorter **??**.

For each project the methodology applied was:

1. Definition of the goals
2. Division of the workload in: PCB Design, Manufacturing and Programing
3. Merge the achievements

2.1.1 PCB Design

In turn, the PCB Design was divided in:

- First circuit sketch
- First selection of the components
- Implementation of the first circuit on a breadboard
- Validation of the design

While in parallel the Eagle work was:

- Design of the schematics' circuit in Eagle
- Design of the board in Eagle. Optimization of the routes and the components' position
- Validation of the design

2.1.2 Manufacturing

Once the design in Eagle was approved, the manufacturing process started. The workload was divided again for two men, trying to optimize the work in parallel, in:

- Components collection
- Cutting of the board
- Printing of the transparencies
- PCB drawing process
- Drilling
- Component's soldering

2.1.3 Programming

A requirement of the project was that the FPGA should serve as the controller for the brick sorting machine which also being linked to a PC via. µTosNet. That requires a hardware description for the unit. The implementation is done in VHDL.

The basis for the implementation is the example provided with the project assignment. The µTosNet is essentially unchanged, except for "our" data being placed on the shared blackboard memory.

Each part is written as an individual component, connected to each other and the physical pins on the top level. The components were each defined as follows:

- Identify component I/O. Ultimately each component should be a "black box" taking an input and generating an appropriate output.
- Write the internal logic of the component.
- Create a test bench and simulate to assert that output is as expected.

2.2 Equipment

To develop and test the designed solution the following equipments were used.

- Oscilloscope
- Multimeter
- Soldering iron
- solder
- FPGA Spartan 3

- Eagle 7.1 light
- Breadboard
- PCB development instruments
- Printer
- DC servo motor
- Slider setup(see picture on front page)
- Colored Lego bricks

Chapter 3

Detecting xy-position of ball impact

This chapter describes the implementation of a solution that enables detection of point of impact for a ping pong ball using an FPGA and piezo electric elements.

3.1 TDOA positioning in a plane

In this section the relationship between differences in time measurements from multiple sensors and the xy-position of the emitter are derived. The problem can be modeled as in figure 3.1 where the following parameters are known from the setup and measurements:

- Time of arrival of sound at sensors a, b, c and d.
- Speed of sound in chosen material.
- Position of sensors.

The system is setup with four sensors since the possibility of resulting in multiple solutions when using only three [**tdoa_book**] The solution will be represented as a system of linear equations to enable fast and relatively simple implementation in VHDL. The derivation will be conducted for two of the sensors giving one row in the equation system. The other two rows are determined in a similar manner.

By assuming a constant speed of sound through the impact plane, equation 3.1 describes the relationship between time difference between measurement and impact $(t_a - t)$ and the distance the impact point is from sensor a.

$$l_a = v(t_a - t) \Leftrightarrow l_a^2 = v^2(t_a - t)^2 \tag{3.1}$$

Equation 3.2 describes the same relationship with respect to sensor b.

$$l_b = v(t_b - t) \Leftrightarrow l_b^2 = v^2(t_b - t)^2 \tag{3.2}$$

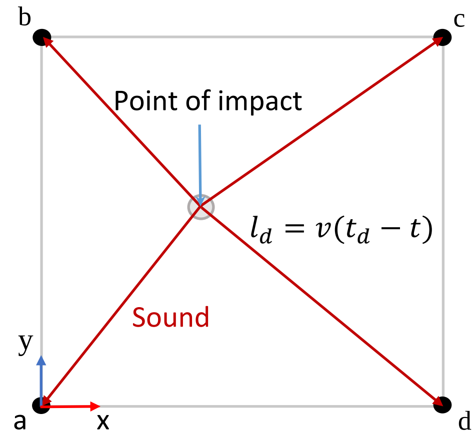

Figure 3.1: Side view of the platform surface area and the net. The red markings are piezoelectric sensors.

By differencing equation 3.1 and 3.2 equation 3.3 can be derived using the difference of squares formula. It is a linear relation with respect to time of impact t.

$$l_a^2 - l_b^2 = v^2(t_a^2 - t_b^2) - 2v^2(t_a - t_b)t \tag{3.3}$$

By using Pythagoras theorem the difference in squared distances from the impact point are also related according to equation 3.4. It can be seen that it is a linear equation in x and y with constants given from the setup.

$$\begin{aligned} l_a^2 - l_b^2 &= (x_a - x_b)^2 + (y_a - y)^2 - ((x_b - x) + (-y_b - y))^2 \\ &= -2((x_a - x_b)x + (y_a - y_b)y) + x_a^2 + y_a^2 - (x_b^2 + y_b^2) \end{aligned} \tag{3.4}$$

By setting equation 3.3 and 3.4 equal and isolating the constants known from time measurements and setup equation 3.5 is derived.

$$(x_a - x_b)x + (y_a - y_b)y - v^2(t_a - t_b)t = (x_a^2 + y_a^2 - (x_b^2 + y_b^2) - v^2(t_a^2 - t_b^2))/2 \equiv k_{ab} \tag{3.5}$$

Using similar relations for the other four sensors results in the system of linear equations 3.6 which solution uniquely defines the xy-position of the impact [**toa_notes**].

$$\begin{bmatrix} x_a - x_b & y_a - y_b & -v^2(t_a - t_b) \\ x_b - x_c & y_b - y_c & -v^2(t_b - t_c) \\ x_c - x_d & y_c - y_d & -v^2(t_c - t_d) \end{bmatrix} \begin{bmatrix} x \\ y \\ t \end{bmatrix} = \begin{bmatrix} k_{ab} \\ k_{bc} \\ k_{cd} \end{bmatrix} \tag{3.6}$$

3.2 Piezo electric elements

This section describes the construction of a number of sensor circuits using piezo electric elements to measure the time of arrival of bending waves. There is a circuit like the one on figure 3.2 for each sensor. It is chosen to make one print for each circuit in order to keep the wires conducting the analog sensor values as small as possible.

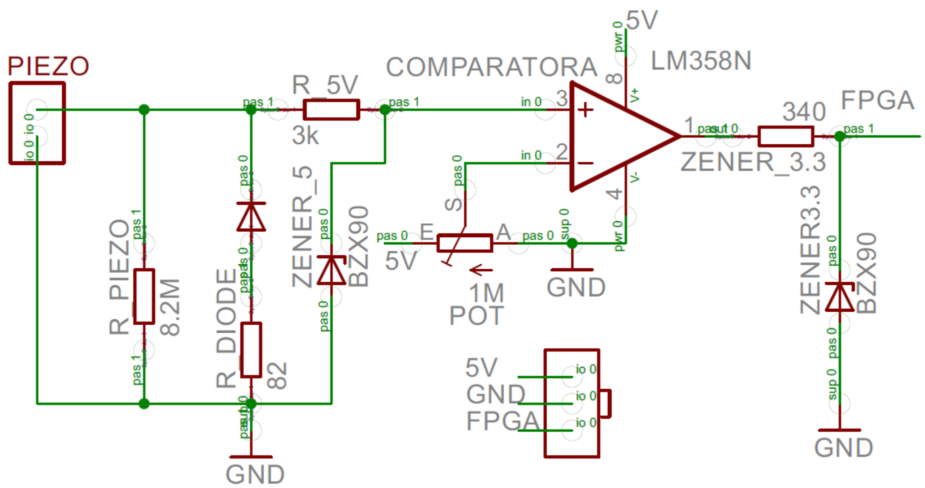

Figure 3.2: Schematics of circuit for digitalizing the output from a piezo electric elements.

The very small current that the piezo electric element generates are amplified with a resistor on $8.5M\Omega$. This value is chosen since test showed that it was possible to detect ball bounce from as low a hight as 30cm. Diodes are used to account for the large positive and negative voltage spikes that the sensor in combination with the large resistor can output in case of a powerful input to the sensor. Since the negative voltage are not use it is removed up to $0.7V$ with a diode which conducts current for ground when the voltage from the piezo becomes less then $0.7V$. The resistor is calculated according to equation 3.4.

$$20 - 0.7V = R_{diode} \cdot 300mA \Leftrightarrow R_{diode} = 64.3\Omega \tag{3.7}$$

Since this resistance was not in stock a resistor of 82Ω is chosen. Where the $20V$ is taken as a guess on the absolute maximum based on the fact that tests has not shown voltages above $15V$ when throwing the ball against the plate. Hence the system are robust against misuse. Since the op-amp can not compare values that are larger than the supply of $5V$, the positive voltage is limited to $5.1V$ with a zener diode [**zener**]. The resistor R_5V is calculated by assuming infinite input resistance in the op amp according to equation

$$20 - 5V = R_{5V} \cdot 5mA \Leftrightarrow R_{diode} = 3\text{k}\Omega \tag{3.8}$$

A zener diode [**zener**] is used to convert the $5V$ on the data output from the operational amplifier to $3.3V$, which is appropriate voltage for the FPGA. The solution with a zener diode is chosen over a another using a voltage divider since the voltage is kept at $3.3V$ for all op-amp of the type LM358 even in case of changes in production. According to the datasheet for LM358 the saturated output voltages varies from the supply voltages down to $1.5V$ below [**lm358**]. The resistor placed on the output of the operational amplifier is calculated using the current used for testing conditions in the datasheet for the diode, as can be seen in equation **??** [**zener**].

$$5 - 3.3V = R_z I_{test} \Leftrightarrow R_z = 1.7V/5mA = 340\Omega \tag{3.9}$$

A potentiometer is connected to the inverting pin of the Op-amp in order to ease the tuning process. It's value is set to 1MΩ to minimize the power consumption.

3.3 Construction of impact plane

The physical platform area will be a flat area of approximately $40 \times 40\,\text{cm}$.

3.4 Solving the TDOA using VHDL

This section describes how the solution for TDOA positioning are implemented in VHDL on a FPGA.

3.4.1 Precise timing

3.4.2 Solving system of linear equations

Chapter 4

Height of ball bounce back

4.1 Determerning energy in impact using ADC

4.2 Bounce back height from measured energy

Derive the kinematics for the ball bouncing back.

Chapter 5

Two joint arm as ball catcher

A arm will be mounted on a stick besides the platform so that the net does not hit the ground. The arm consist of two revolute joint enabling it to place the net on any xy-coordinate above the platform.

5.1 Physical construction

How the arm is constructed to be robust and fast while being able to reach the specified area for impact.

5.2 Inverse kinematics and control

How the arm are controlled to a specified xy-position

Chapter 6

PC Program

A PC program will show the position of the hit point and the hight the ball will bounce back to.

6.1 Communication and Logic

Regarding the communication, the PC program and the FPGA are connected by a USB wire and this protocol is handle by the qt serial libraries given. This let the communication in hands of a code that has been already tested.

In respect of the logic behind the PC program, the PC ask to the FPGA if a new brick has been detected, if so, ask for the color of the new brick. This is implemented thanks to two registers in the FPGA, one for the *newbrick* that turns true when a new brick is detected, and another for the *color* where the values of the brick's color is saved.

Also, the detection thresholds of the new brick can be change from the PC interface so the FPGA hasn't had to be reconfigured.

6.2 Interface

The main reason to implement a UI is the easy to introduce the new values for calibration. Due to the system is portable and the environment changeable, the system require of some values to be modifiable in order to calibrate it. In this case there are two factors to modify: the refresh and the thresholds.

On one hand, the refresh determines when the PC program ask to the FPGA for a new brick. A higher value will decrease the total power consumption while it could mean that, if a new brick appears before the order was sent, a new brick can be lost by the PC program.

On the other hand, the thresholds for each brick need to be calibrated depending on the ambient conditions. Despite the fact probably this only need to be made once, due to the fact the system is under controlled conditions, an easy calibration system has been implemented as shows the figure 6.1.

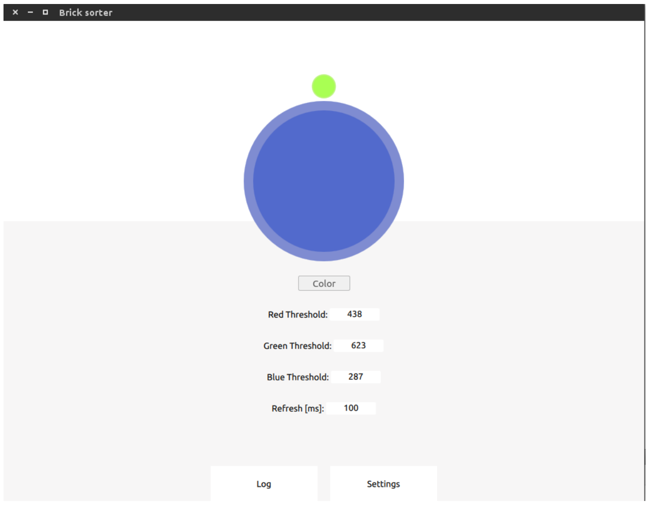

Figure 6.1: PC program implemented with Qt. Detail of Settings tab.

Furthermore, the PC program has a data log where relevant information of the FPGA is shown. The UI also shows the last brick detected by the machine and shows a graphical history in form of smaller circles as can be seen in the figure 6.2.

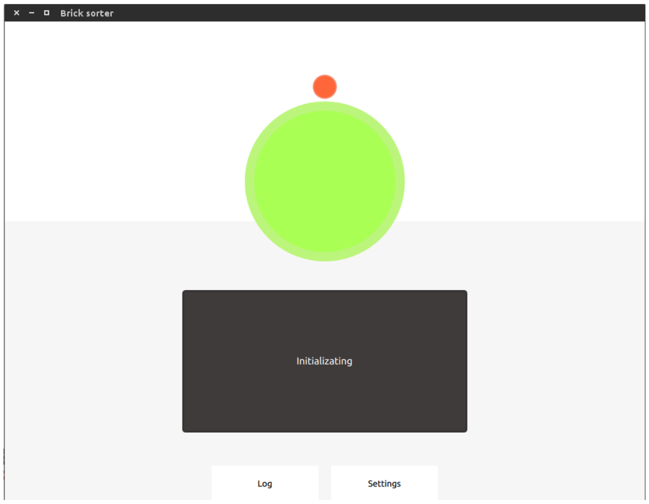

Figure 6.2: PC program implemented with Qt. Detail of Log tab.

Chapter 7

Conclusions

Sum up of results.