**Bin Knows Best**

**Andrei Sabau**

**sabau.andrei14@yahoo.com**

**Andrei-Cristian Baraian**

**andreibaraian88@gmail.com**

**Victor Oltean**

**oltean.victor10@gmail.com**

**Submitted for the 2017 Digilent Design Contest Europe**

**13th of May 2017**

**Advisor: Dr. Eng. Mihai Negru**

**Technical University of Cluj-Napoca**

**Cluj-Napoca, Romania**

|  |
| --- |
| Link to the presentation video: https://www.youtube.com/watch?v=AdwuJbIZPsY&yt%3Acc=on |

[**Introduction**](#_qxevqcvpyrtv) **4**

[**Background**](#_z3o7w7rceifq) **7**

[Design](#_7luvei89y0zb) 8

[Design Overview](#_ehd2m7joeamw) 8

[Detailed Design Description](#_cep5knxbcr93) 9

[Zybo Board](#_w9ea7a8yrqw1) 10

[Physical Trash Bin](#_ohhu4p4ts7k1) 10

[Infrared Transparency Detector](#_63s6q3fwl3r6) 10

[Metal Detector](#_c1u5w5gw0erz) 11

[Weight Sensor](#_gsajmrugxwld) 14

[Distance detector](#_he2yybswes54) 18

[Servo motor](#_9978camxhj0p) 19

[Bin compartment](#_xvh4b3mhnxpw) 21

[**Components**](#_5zrmgvybsrpg) **21**

[Mono pulse generator](#_3t6s53pu1mn5) 21

[HX711](#_bx1vq3ni63yu) 23

[Generating SCK](#_asu88npf68qr) 23

[Reading the Data](#_f3nyqi4kxc9c) 23

[State Machine](#_p468m5lbnxuy) 24

[Ultrasonic](#_xitehtvfojfe) 25

[ServoX (X = 1, 2, 3, 4, 5)](#_avlegiz3p95p) 26

[Servo Control](#_pc3ep9em8czr) 27

[Main](#_xlp4g0i338ji) 29

[**Discussion**](#_u308n7gjs69) **33**

[Problems Encountered](#_yczwjsru3qf) 33

[Marketability](#_29swh7dios6w) 33

[Community Feedback](#_whi3amwmwyqz) 33

[**References**](#_puaxhzobq9zy) **34**

[**Appendix A**](#_szzwwoo4h2r5) **34**

# 

# 

# Introduction

**Abstract**

This report represents a contest entry for the 13th edition of Digilent Design Contest and presents the design of an autonomous sorting trash bin, with the name “Bin Knows Best”. The system is capable of analyzing an object and directing it to its corresponding compartment. The analysis is performed based on various inputs collected from multiple sensors, and the result is the type of material that object is made of, i.e., plastic, metal, glass or undetermined (waste). Because of the smart design, i.e., the four compartment doors form a bottom up pyramid shape, after the item was processed in a processing room, it falls from the scanning room into the desired compartment by opening a door with a servo, and keeping the other three doors closed, so that these three compartment doors will direct the falling object to its place decided by the design.

The system is also capable of detecting when one of the bins is full and generates a warning signal, displayed on an LCD, until that bin is emptied. On the LCD, which is mounted on the top of the design, the user can also see if the system is currently busy scanning an object or if it is in its idle state, waiting for an item to be introduced.

As a safety feature, it the “start” button is pressed, but no object was introduced, the system will remain in its idle state, waiting for an item to be introduced.

The project was developed using Vivado WebPack 2016.4, Visual Studio 2015 Community Edition and Arduino IDE.

**Objectives**

The objective of our project was to create a fully autonomous and independent system for detecting various types of objects, and classifying them based on the materials they are made of, such as plastic, metal and glass, with the purpose of creating an efficient solution for the recycling challenges encountered nowadays. Also, our target was to reduce as much as possible the item’s analysis time and provide a fast and correct classification.

Moreover, our proposed model was designed as a modular architecture in order to ensure flexibility when trying to update the system or scale it up. As a result, if a better approach for detecting a type of object would be discovered, it would be easily integrated and implemented in our project, without requiring a lot of modifications to be performed and unnecessary overhead.

Our intention was to offer a small sized system as a proof of concept, which can be integrated in homes or offices or it can be further extended to solve problems at larger scales.

**Project Summary**

Our design proposes a better approach to what recycling means and how it is implemented in the modern world. The smart trash can is able to detect an object, analyze it and classify it correctly and then route it to the designated compartment with no human intervention.

When the object is placed in the processing room, the sensors already collect information about it, and when the user presses the button to notify the system that an item was introduced and needs to be processed, the design takes data from the sensors and determines what type of object was introduced and where it should be placed. Our project has 4 compartments, representing some of the most important materials that need to be recycled, such as metal, glass, plastic and if the system could not determine the type of an item it will regard it as general waste.

The sensors’ room is composed of a trapdoor and sensors. When an object is introduced in the scanning room, it is placed on the trapdoor, so that after the item is analyzed, the trapdoor is lowered down allowing the item to fall to one of the compartments. In the processing room, there are 3 types of sensors, used by the system to distinguish between materials:

* a metal detector: it was created by our team, by following the simple principles of regular metal detectors used for searching coins or valuable metals in the ground
* two infrared sensors: the infrared sensors are two pairs of phototransistors and infrared light emitting diodes (LEDs) which detect if an object from the processing room is transparent or not
* a weight sensor: it is composed of a one kg load cell and an HX711 integrated circuit that detects the changes in the resistance of the load cell, amplifies the values and converts the analog signals into digital ones

Using these sensors the system follows a path to determine if an object is made of metal, plastic, glass or if it is a regular trash object. The path pursued by the design is the following one: it checks if the item is a metal object or not; if it is a metal, the processing of the object stops and the item is directed to the metal compartment, otherwise, it checks if the object is transparent or not; if the item is not transparent it will be regarded as waste (regular trash) and will be directed to the waste compartment, but if the item from the processing room is transparent one final check is performed on the object, i.e., if the item is heavier than a certain threshold, in our case 100g, the object is seen by the system as being made of plastic material, otherwise, as being made of glass; after this final check, the item is placed in the corresponding compartment: glass compartment or plastic compartment.

In order for the above path to work and give a correct result, our system introduces the following constraints:

* the height of an object should be 20 cm ± 10 cm, because the height of the processing room is 20 cm and the coil that detects metal is placed on the top of the room, so if a small metallic item is introduced it might not be detected as being made of metal
* the glass and plastic objects need to be transparent
* the volume of the transparent objects should be approximately the same and they should be empty, because when the system tries to differentiate between glass and plastic, it does this by weight, so if the items have the same volume and are not filled up with anything ⇒ a glass item will always be heavier than a plastic object

The system is only a prototype and that is why it introduces the above mentioned constraints. It tries to solve only the most common cases, the particular ones requiring more logic, sensors and resources.

After the object was identified, the system simultaneously lowers the trapdoor and opens one compartment door corresponding to the identified type of material the item is made of. Because the other three compartment doors are closed, when the object falls from the processing room they will direct the falling item to pass through the open door and end in the corresponding compartment.

Another feature of the system is that whenever an ultrasonic sensor detects that one bin is full, i.e., the objects from a compartment are at the top of that bin, the users are notified through an Liquid Crystal Display and the system is blocked, meaning that it will not accept any more objects to be analysed until that specific bin is emptied.

**Hardware Requirements**

**Digilent Products Required**

The following hardware was provided by Digilent:

1. Zybo Zynq-7000 ARM/FPGA SoC Trainer Board
2. GWS Servo: S03TXF STD (5)
3. Servo Mounting Bracket (5)
4. Pmod WiFi: WiFi Interface
5. Pmod GPS: GPS Receiver
6. Pmod AD1

In combination to the Digilent hardware, we have also used:

1. Arduino Uno, Arduino Nano
2. LCD Shield
3. Ultrasonic Sensor - HC-SR04 (4)
4. Level shifter 3.3V-5V (3)
5. Infrared LED (2)
6. Phototransistor (2)
7. 1 kg load cell
8. HX711 I.C.
9. Copper coil
10. Resistors, transistors and capacitors
11. Wires

The number of sensors are given just as a reference, depending on the particular need, these numbers may be subject of change.

**Software Requirements**

1. Vivado HLx Design Suite 2016.4
2. Visual Studio 2015 Community
3. Notepad++ (more editing options than Vivado)
4. Arduino IDE

**Tools Required**

During the development of the project, the following tools were required:

1. Multimeter
2. Soldering iron
3. Welding electrodes
4. Grinder
5. Cardboard
6. Screws
7. Metal bars

**Design Status**

The design is currently in an almost complete stage, being able to solve the problems it was designed for, i.e., it correctly sorts various kinds of trash objects, even though they have to respect certain constraints, but lacks a couple of features which were supposed to be part of the project, such as trash bin location detection, implemented with the Pmod GPS and communication through the internet, which was supposed to be done with the Pmod WiFi. These features would have been used in the following scenario: if a group of trash bins were to be deployed into multiple locations, the administrators would have needed to know in real-time when the compartments of a system were full, in order to empty them as fast as possible and unblock the machines. The solution to this issue would have been that each ”Bin Knows Best” system would have sent its location through the internet whenever one of its compartments were full.

# Background

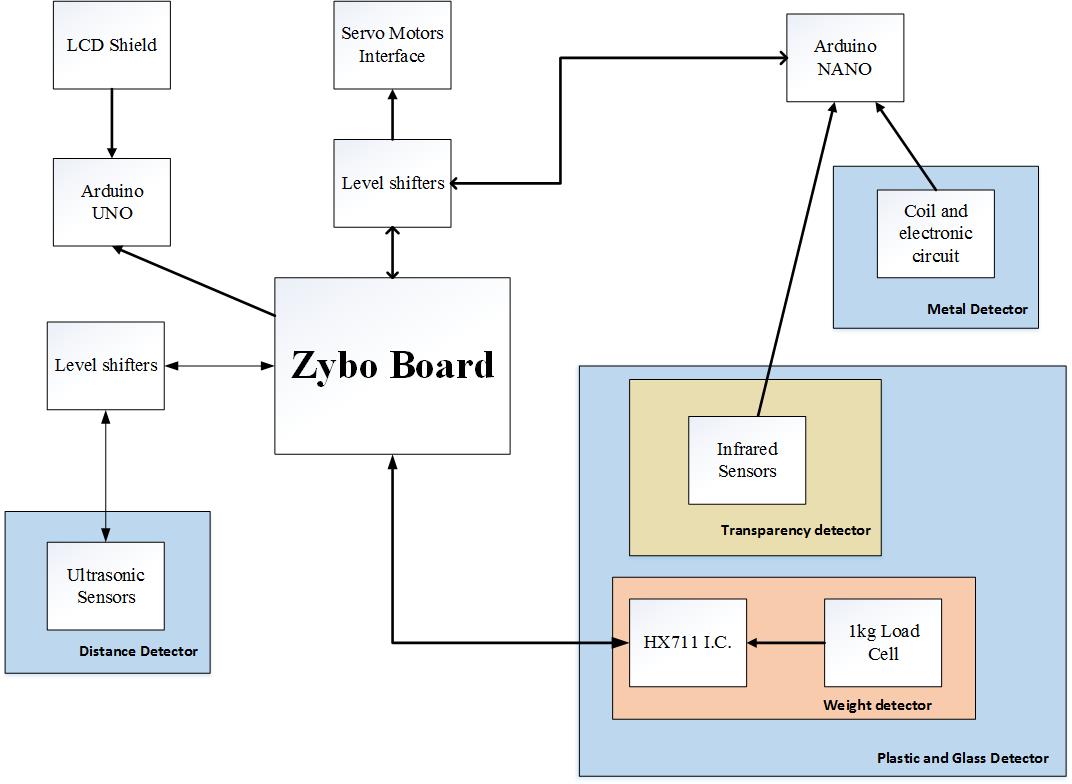
**Why This Project?**

Living in a modern day that is characterized by consumerism, the problem of recycling has just recently emerged, but the undeniable fact is that it needs immediate attention and efficient solutions to be developed. The sorting of trash is a very complicated problem and there are no straightforward sensors to deal with this issue. This is a problem common for big factories that process tons of garbage, but we want to focus on the local scale, because if every human would recycle, the world would be greener and less resources would be wasted. In recent years, the communities have set up different types of bins for people to help classify and recycle the garbage. Theoretically, the idea is a very good one, but as studies showed, many people tend to ignore them, or throw their garbage wherever is most convenient, not keeping in mind where the actual place of that garbage is.

This is why we have came up with the idea of designing a smart trash bin which is able to sort the garbage, without requiring people to do extra things other than throwing the objects in the trash. This way, (ideally) no error would occur in the process of sorting since it is automated. Moreover, because of the modularity of the project, it can be easily extended to serve a large scale purpose and represent a viable product for the modern market. We need to see recycling not as a trend, but as a major solution to the climate problems of the 21st century. What we wanted was to design such a project to help the planet overcome these current problems.

## Design

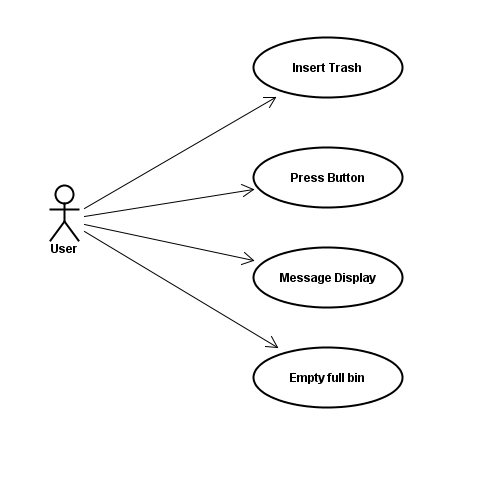
### Design Overview



### Detailed Design Description

This chapter is going to present in detail each part of our project, following a top-down approach for a better understanding of the system.

To begin with, below is a use case diagram of our project. The fact that the use case is not so rich is because we want as few interactions ‘human-machine’ as possible, since the system is supposed to be autonomous and is also intended to provide no extra work for people, other than throwing the trash in the bin.



The main processing unit of the project is the Zybo board, which collects data from the sensors. Co-operating with the Zybo board is an Arduino Nano board and an Arduino Uno board, one used for collecting and processing data from the infrared sensors and the metal detector, and the other one is used for interfacing with an LCD screen which displays relevant messages for the users of the system.

### Zybo Board

The main control of our project is the Zybo (Zynq board), which is a feature-rich, ready-to-use, entry-level embedded software and digital circuit development platform. Because of the various I/O PMOD, it represents a powerful central control unit, capable of processing multiple inputs and sending multiple outputs. Because we wanted to get full control of the interfaces and of sensors in order to better understand how they work, the whole logic is implemented based on the FPGA Programmable Logic. To be mentioned that the Zybo board also has an ARM Processing System which in combination with the FPGA offers a very powerful tool for any hardware programmer. For further implementations on the project, it is clear that an integration of the ARM would be essential.

### Physical Trash Bin

The trash bin is composed of three main parts:

* the scanning room - place where the analysis of the object is performed
* bottom up pyramid - pyramidic shape made of the doors corresponding to each compartment
* four trash compartments - the compartments store: plastic, glass, metal and waste

### 

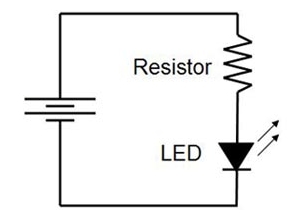
### Infrared Transparency Detector

Infrared light passes through transparent objects. We use this property in order to differentiate plastic and glass items from other types of objects.

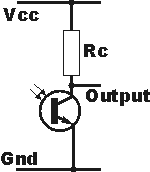
The circuit is composed from two parts:

* infrared emitter
* infrared receiver

The emitter works just like a normal LED, having a resistance of 220 Ohms.



The receiver, however is actually a phototransistor, having the following circuit connection

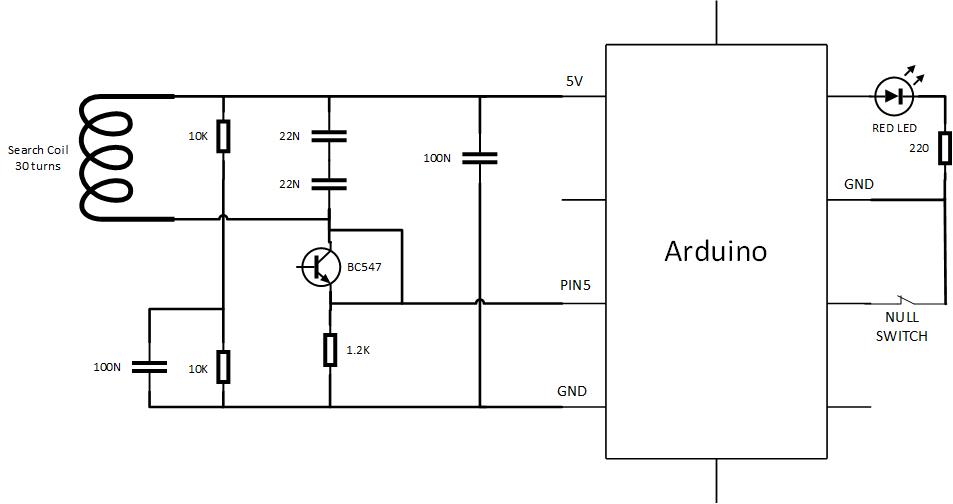


The configuration is similar to the one of a voltage divider. The emitter is pulled to VCC via a resistor.

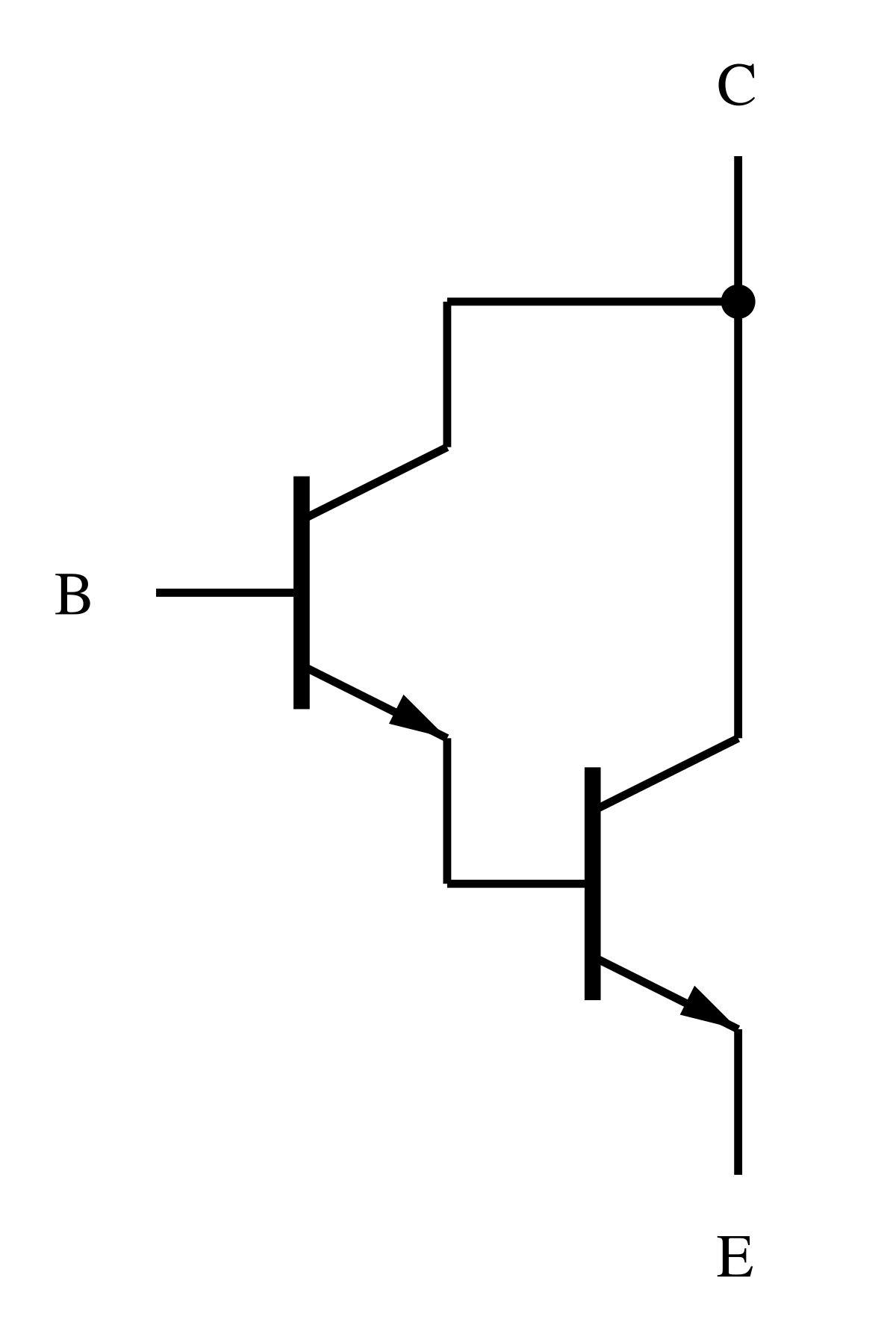
The output is connected to an analog pin from the Arduino Nano. As more infrared light is detected by the receiver, it allows more current to pass through it, thus making its equivalent resistance lower. Ideally, if the light from the emitter would pass unobstructed to the receiver, the voltage read on the analog pin of the Arduino board would be 0V.

In the scanning room, we have two such pairs of circuits, making sure that the object is detected by at least one pair.

### Metal Detector

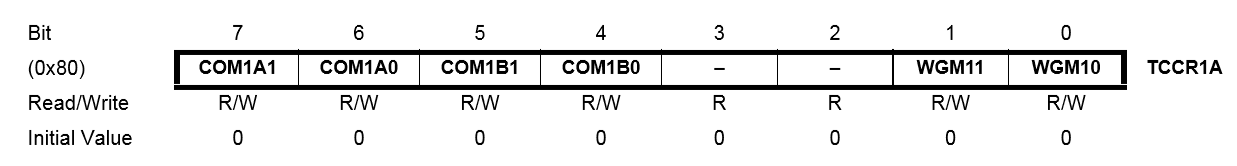
When designing the metal detector, the research carried by us resulted that there is no straightforward, low-cost sensor which can detect if an object is made of metal or not. The available sensors on the market are industrial inductive sensors which detect metal objects without requiring physical contact with the target to be sensed. The major drawback of these sensors is that they are very expensive. Thanks to the open-source community of robotics, we were able to find out how to design our own metal detector. Following this [link](http://dzlsevilgeniuslair.blogspot.ro/2013/07/diy-arduino-based-metal-detector.html), we understood the basic principles of building a metal detector. The basic idea is that metal detectors use a coil as an oscillator circuit. Whenever a metal is in the proximity of it, it enters the magnetic field created by the inductor. This results in the change of magnetic permeability which causes the inductance to change. Such a change will modify the oscillating frequency of the coil, signaling the presence of the metal. Above is the schematic of the oscillating circuit.

In practice, metal detectors are build using a beat frequency oscillator (BFO), but a viable alternative is also a microcontroller, like the Arduino. The circuit oscillates at about 160 kHz and sends the signal to pin 5 of the microcontroller. The components of the circuit are: two resistors of 10k Ohm, 1 resistor of 1.2k Ohm, 2 capacitors of 22nF, 2 capacitors of 100nF, a BC547 transistor and a copper wire used as coil with 30 turns. Initially, we tried to use a 2n2222 transistor as stated in the blog, but we could not make the circuit work, so we switched to the BC547 one. Another option was to place 2 x 2n2222 in Darlington configuration.

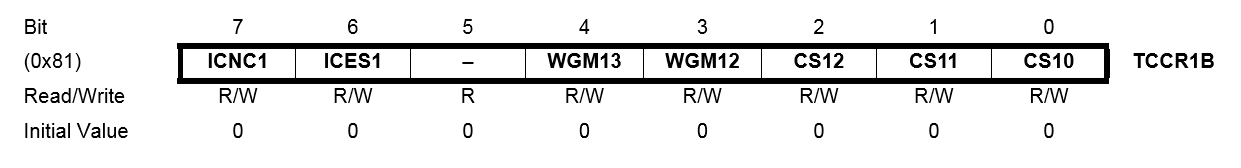


Darlington configuration

Regarding the software part of the metal detector, the signal fed to the pin5 was used as an external clock in order to keep track of the detector’s oscillation frequency. For this task, there are no pre-defined libraries capable of doing this, so there was the need to configure the timer/counter registers (TCCR) through TCCR1A and TCCR1B.

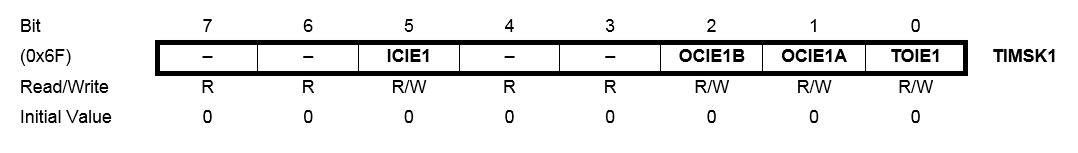


In the Arduino sketch, we set TCCR1A = 0. This means that the system will have a normal port operation, OC1A/OC1B being disconnected (asserted by COM1A1 and COM1A0). In the same way the Compare Output Mode for Channel B will be asserted. The last two bits mean that the counter mode operation will be normal, counting to the maximum (0xFFFF).



Next, the register TCCR1B = 0x07. Starting from the Most Significant Bit:

* ICNC1 set to 0 means that Input Capture Noise Canceler is not activated, therefore not filtering the signal.
* The ICES1 bit is used for choosing the edge on the Input Capture pin that is used to trigger a capture event. In our case it is set to 0 and means that a falling edge is used as a trigger.
* Bit 5 is reserved and should be written to zero when TCCR1B is written.
* Bits 4-3 represent the Waveform Generation Mode and they are used in normal operation mode, counting to 0xFFFF.
* The three LSB represent the Clock Select, selecting the clock used by the timer/counter. We need to set each of them on 1, in order to select the T1 pin as external clock source. Basically, this clock will be the one from the oscillating circuit.

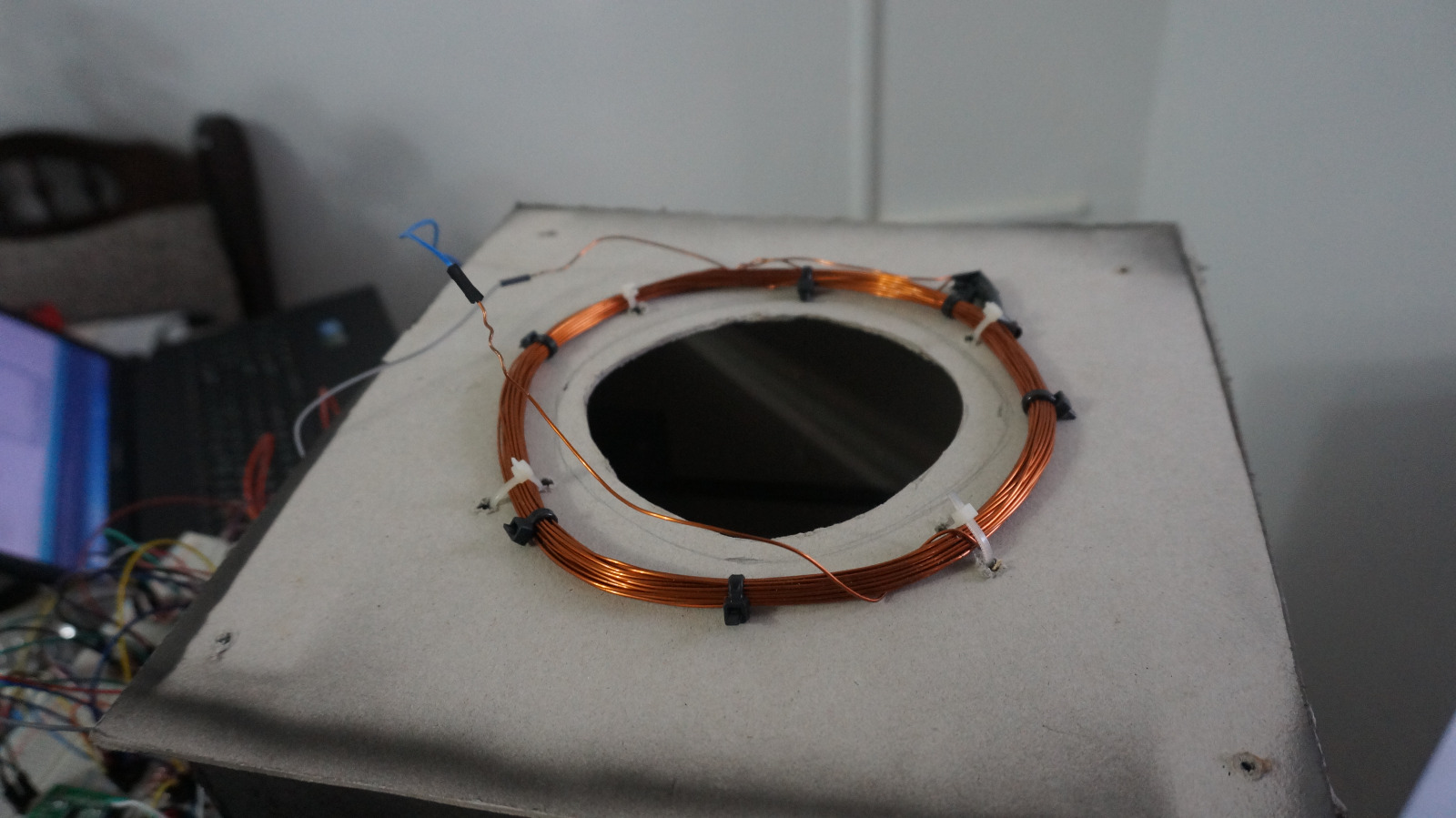


TIMSK1 represents the Timer/Counter1 Interrupt Mask Register. Using the SET operation, we make an OR with TIMSK1 register and the OCF1A flag, setting the OC1E1A bit. When the OC1E1A bit is one and the I-flag in the Status Register is set, the Timer/Counter1 Output Compare A Match interrupt is enabled.

The interrupt function SIGNAL(TIMER1\_COMPA\_vect) keeps track of the number of microseconds elapsed since the last time the function was called. For this function, we use the OCR1A 16-bit output compare register. Its value is always compared with the Timer/Counter value all the time. Basically, in our case, the result will set the compare match flag (OCF1A) which is used to generate an output compare interrupt request. In our code, we generate interrupts every 100 oscillations of the search coil.

When sampling a new null value reference for the oscillations, in addition to the manual button, we also added a signal transmitted by the Zybo board every time when the finite state machine of the design reaches the set\_up state. Therefore, each time a new process starts, we will calibrate the metal detector sensor. Moreover, the code allows for easy adjustments on the sensitivity of the detector. For debugging purposes, we used two LEDs: the first Yellow LED signals the frequency of the oscillating circuit transmitted by the Arduino as a series of Geiger counter clicks. The Red LED acts as a boolean value, signaling if the object is a metal or not.

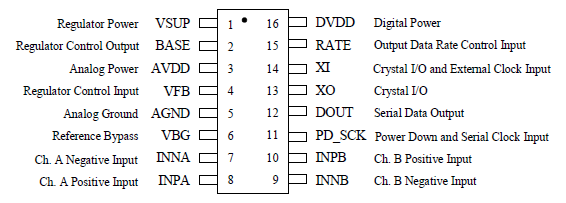
In order to integrate the coil in the design, the best idea was to place the coil directly at the entrance in the processing room. This way, any object would be right in the magnetic field and would change it (or not) depending on its nature.



### Weight Sensor

**HX711**

The HX711 integrated circuit is a 24-bit analog-to-digital converter (ADC) designed for weighing scales and industrial control applications to interface directly with a bridge sensor, and it has two selectable differential input channels (channel A and channel B). The block diagram of the chip is shown below:



**Load Cell Amplifier**

The Load Cell Amplifier is a board for the HX711 IC, allowing users to easily read load cells and measure weight. It does this by taking care of more than half of the pins of the HX711 IC, so that the users will only work with two pins from the integrated circuit for data retrieval, i.e., DOUT and PD\_SCK. The two pins that form the two wire serial interface for communication are also used for data input selection, gain selection and power down controls. When output data is not ready for retrieval, digital output pin DOUT is high. Serial clock input PD\_SCK should be low. When DOUT goes to low, it indicates data is ready for retrieval. By applying 25~27 positive clock pulses (depending on which channel is used) at the PD\_SCK pin, data is shifted out from the DOUT output pin. Each PD\_SCK pulse shifts out one bit, starting with the Most Significant Bit first, until all 24 bits are shifted out. The 25th pulse at PD\_SCK input will pull DOUT pin back to high.

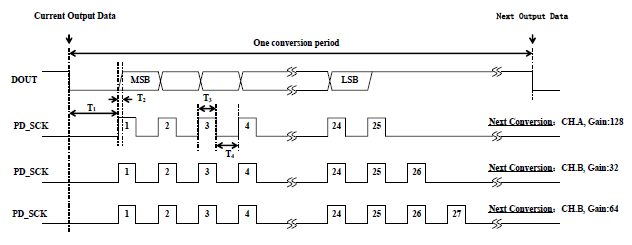
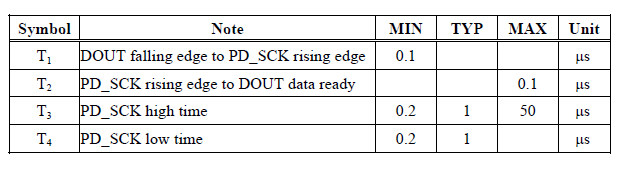


figure showing the timing diagram for retrieving data from the HX711 chip



Input and gain selection is controlled by the number of the input PD\_SCK pulses. PD\_SCK clock pulses should not be less than 25 or more than 27 within one conversion period, to avoid causing a serial communication error.

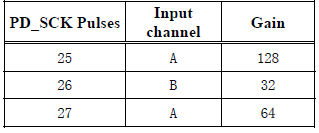
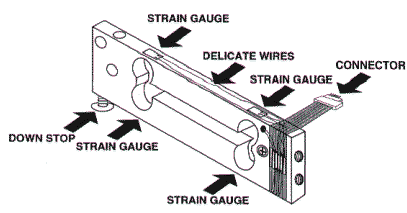


table representing channel and gain selection

**Load cell**

A load cell is a physical element (transducer) that can translate pressure (force) into an electrical signal.

In the project we have used a strain gauge load cell like in the picture below:



The strain gauge load cell is a mechanical element which detects the force applied on it by the deformation of several, in our case four, strain gauges that are placed on the load cell.

Bar strain gauge load cells, are set up in a “Z” formation so that the torque is applied to the bar and the four strain gauges on the cell will measure the bending distortion, two measuring compression and two tension. When these four strain gauges are set up in a Wheatstone bridge formation, the small changes in resistance (from μΩ to mΩ) from the strain gauges are accurately measured.

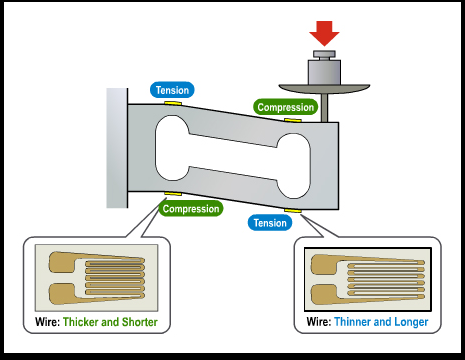
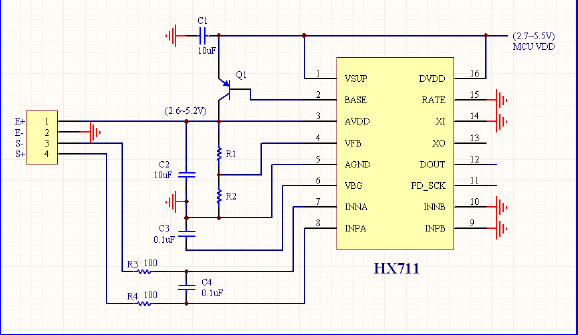


figure showing the Z formation a load cell is placed in and the corresponding four strain gauges

The schematic of the Load Cell Amplifier board (HX711 chip and the additional electric circuits) connected to a load cell is presented below:



### Distance detector

For detecting the distance we have used an Ultrasonic Ranging Module HC - SR04. This sensor provides a 2 cm up to 400cm non-contact measurement function, with the range of accuracy reaching approximately 3mm.

The module used in this project includes an ultrasonic transmitter, receiver and a control circuit. The basic principle of functioning is the next one:

* Using the I/O trigger, HIGH logic level signal is sent for at least 10 μs
* After the 10 μs HIGH logic level signal the module automatically sends eight 40 kHz signals and detects if there is a pulse signal back (echo)
* If a signal back is detected, its HIGH logic level state is counted. This HIGH logic level state represents the time passed between the eight 40 kHz ultrasonic signals and the received signal (echo signal)

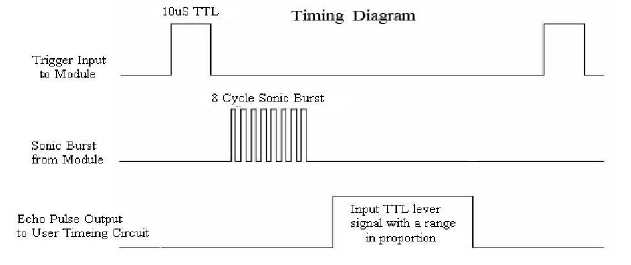


figure representing the timing diagram of the principle of functioning of the ultrasonic module

The distance is calculated based on the time interval between the sent trigger signal and received echo signal. To calculate the distance in centimeters the next formula was used:

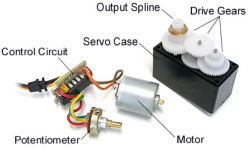
**μS / 58 = centimeters;** μS represents the time the echo signal was HIGH

The speed of sound is 340 m/s or 29 microseconds per centimeter. The ultrasonic waves travel from the module to the object and back, so the distance from the module to the object is half of the distance travelled by the ultrasonic waves : μS / (29 / 2) = μS / 58.

### 

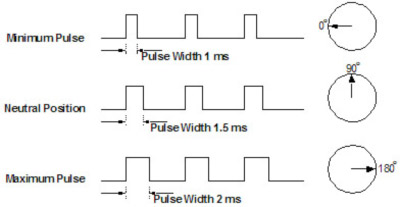
### Servo motor

A servo motor is composed of a small DC motor, a potentiometer and a control circuit. The motor is attached by gears to the control wheel. As the motor rotates, the potentiometer's resistance changes, so the control circuit can precisely regulate how much movement there is, and in which direction.



When the shaft of the motor is at the desired position,power supplied to the motor is stopped. If not, the motor is turned in the appropriate direction. The desired position is sent via electrical pulses through thesignal wire. The servo motors are proportional control systems, meaning that the motor's speed is proportional to the difference between its actual position and desired position, so if the motor is near the desired position, it will turn slowly, otherwise it will turn fast.

Servos are controlled by sending an electrical pulse of variable width, also known as a PWM signal, through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movement. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise and counter-clockwise direction. The PWM sent to themotor determines the position of the shaft, and based on the duration of the pulse sent via the control wire, therotor will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds and the length of the pulse will determine how far the motor turns. For example, a 1.5 ms pulse will make the motor turn to the 90° position. A pulse shorter than 1.5ms moves the servo in the counter-clockwise direction toward the 0° position, and any pulse longer than 1.5 ms will turn the servo in a clockwise direction toward the 180° position.



All the servo motors are different from one another, so in order to move the servos from our project to 0°, a 0.5 ms pulse needs to be sent, and in order to move them to 180°, a 2.25 ms or 2.5 ms pulse needs to be sent through the control wire.

### 

### Bin compartment

The ‘Bin Knows Best’ system has 4 compartments which are made in a triangular prism shape, like in the picture below:

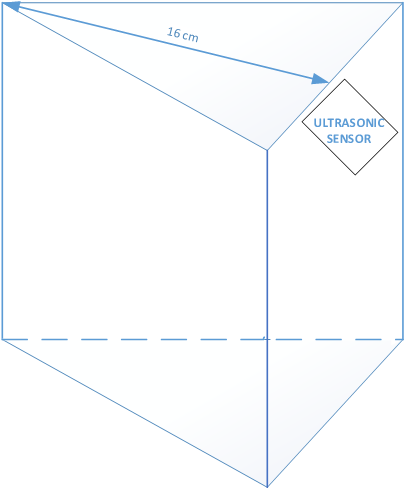


figure representing the shape of a compartment

The dimension of the of the compartments are: height - 35 cm, base - isosceles triangle having two edges of 25 cm and one of 37 cm. Because of this, the height of the isosceles triangle is 16 cm, so the ultrasonic sensor placed below the top of the compartment, on the longest edge of the triangle will always detect approximately 16 cm, except when an object is blocking its view towards the corner. This is when our system interprets this result as a warning that a certain bin is full, leading to the blocking of the design until that specific bin was emptied.

# 

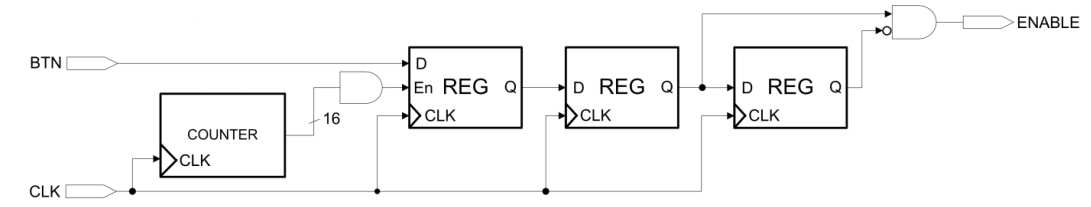
# Components

## Mono pulse generator



The mono pulse generator component has the role of generating an enable signal once per button push. If this component was not used, then the system would sense that the button was pushed many times, since they are not ideal buttons.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Direction** | **Description** |
| CLK | 1 | Input | 125MHz, provided by the ZYBO board |
| btn | 4 | Input | The 4 buttons from the ZYBO board |
| enable | 4 | Output | The 4 mono pulses |



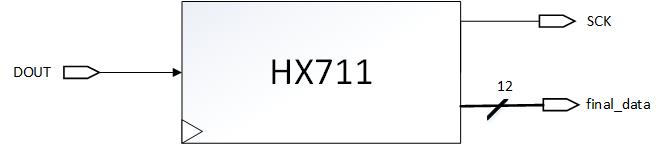
The role of the first register together with the counter is to offer a delay large enough for correctly debouncing the buttons. The size of the register may be increased if the buttons are in a worse state.

To be noted that the mpg unit was not used for example when trying to send a pushed button to the Arduino. This is because that way the mpg sends a high signal only one clock cycle, being too little for the Arduino board to detect it. Routing the button directly to the output to the Arduino, we can send the signal for much more clock cycle, being able to detect the pushed button.

## 

## HX711

The HX711.vhd component is used in order to read data serially from the weight sensor. The data received will be on 24 bits. However, the 12 least significant bits are disregarded because for our application we have determined that the 12 most significant bits can determine the weight of the object at gram scale.



Signals:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Direction** | **Description** |
| CLK | 1 | Input | 125MHz provided by the ZYBO board |
| DOUT | 1 | Input | the data from the HX711, read bit by bit |
| SCK | 1 | Output | a clock generated by this component, having a frequency of 500KHz. |
| final\_data | 12 | Output | the 12 most significant bits read from the weight sensor. |

### Generating SCK

The SCK signal needed by the HX711 in order to send data should have a period of 2 microseconds: 1us HIGH and 1 us LOW. That translates to a frequency of 500 KHz. The ZYBO board’s internal clock works at a frequency of 125MHz. Therefore, the clock needs to be divided by 250 in order to make the communication.

### Reading the Data

The data from the sensor is read during the falling edge of the SCK, because we are sure that at that moment in time, the bit is ready to be read. In order to detect the falling edge, a 2 bit shift register is used, containing the previous SCK value and the current SCK value. “10” in the register translates to a falling edge, and, therefore, we can read the current bit from the DOUT input.

The read bits come MSB first and are stored in a register. After reading 24 bits, the state is changed to IDLE.

### State Machine

|  |  |
| --- | --- |
|  | The implementation of this component is a Finite State Machine, having the following states: INIT, START, WAIT\_BEFORE\_READ, READ\_DATA.  In the INIT state, the SCK is pulled LOW. The transition to the START state begins as the weight sensor pulls the DOUT line LOW.  The state WAIT\_BEFORE\_READ is necessary, because the HX711 requires a 0.1 microseconds wait time before sending the SCK signal.  The READ\_DATA state goes on for 6250 125MHz clock cycles. That is because for each bit we have 250 cycles, and we read 24 bits. The last 250 cycles are used for sending the last clock bit, signaling the IC to stop.  In this state, the SCK is connected to the 500KHz clock, and the data read stored in a shift register. |

## 

## 

## Ultrasonic

The ultrasonic.vhd entity is used to communicate with the ultrasonic modules used in the project.



Signals:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Direction** | **Description** |
| echo | 1 | Input | signal received by the board after the trigger signal was sent |
| CLK | 1 | Input | 125MHz clock provided by the Zybo board |
| trig | 1 | Output | trigger signal with 10 μs HIGH level that will be sent to the ultrasonic module |
| less\_than\_16cm | 1 | Output | signal that represents the status, i.e., full (‘1’) or not (‘0’) of one compartment |

The entity generates a triggering signal that has a HIGH (‘1’) logic level for 10 μs, using a 100 kHz clock, and then waits to receive the echo signal. If an echo signal is received, its HIGH state is counted using the 100 kHz clock. After the echo signal was fully received the distance from the ultrasonic module to the object is calculated and the ‘less\_than\_16cm’ 4-bit signal is computed: if the calculated distance is less than 16 cm on a certain position of the ‘less\_than\_16cm’ vector, a ‘1’ will be placed, otherwise a ‘0’ will be placed.

## 

## ServoX (X = 1, 2, 3, 4, 5)

We have used X, because we use five servo motors in our project and the entity-architecture files are the same for everyone.



|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Direction** | **Description** |
| CLK | 1 | Input | 125MHz provided by the ZYBO board |
| move | 2 | Input | signal controlling how a servo should move |
| servo\_sig | 1 | Output | PWM signal controlling the servo |

The servo entity creates internally a 4 kHz clock that has a 0.25 ms period and 50% duty cycle, because the PWM signal created will have a 20ms period, and that period will be in a HIGH logic level for 0.5 to 2.5 ms, depending on what direction we want the servo to be moved on.

The ‘move’ signal can be:

* “00” or “11” - the servo does nothing
* “10” - the servo will move to 180°
* “01” - the servo will move to 0°

Depending on the value of the move input signal, this component creates the PWM signal for the servo motor and sends it through the ‘servo\_sig’ output.

## Servo Control

In this component, the signals for each individual servo motor are set.



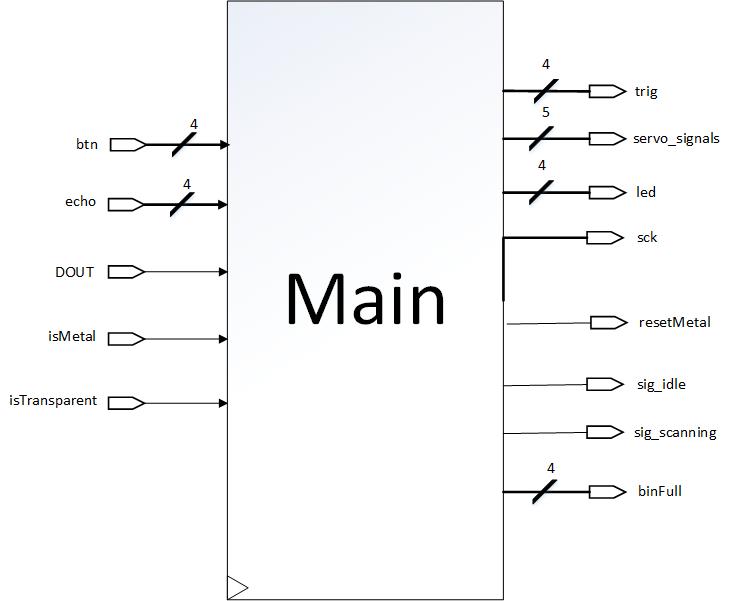
|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Direction** | **Description** |
| CLK | 1 | Input | 125MHz provided by the ZYBO board |
| servo\_status | 5 | Input | Specifies which servos should move. |
| servo\_signals | 5 | Output | The actual signals sent to the servos |

In this component, the ServoX subcomponents are instantiated. The servo\_signals output is actually a vector created from the 5 “servo\_sig” signals from ServoX. This component also sends the move signals to each ServoX component.

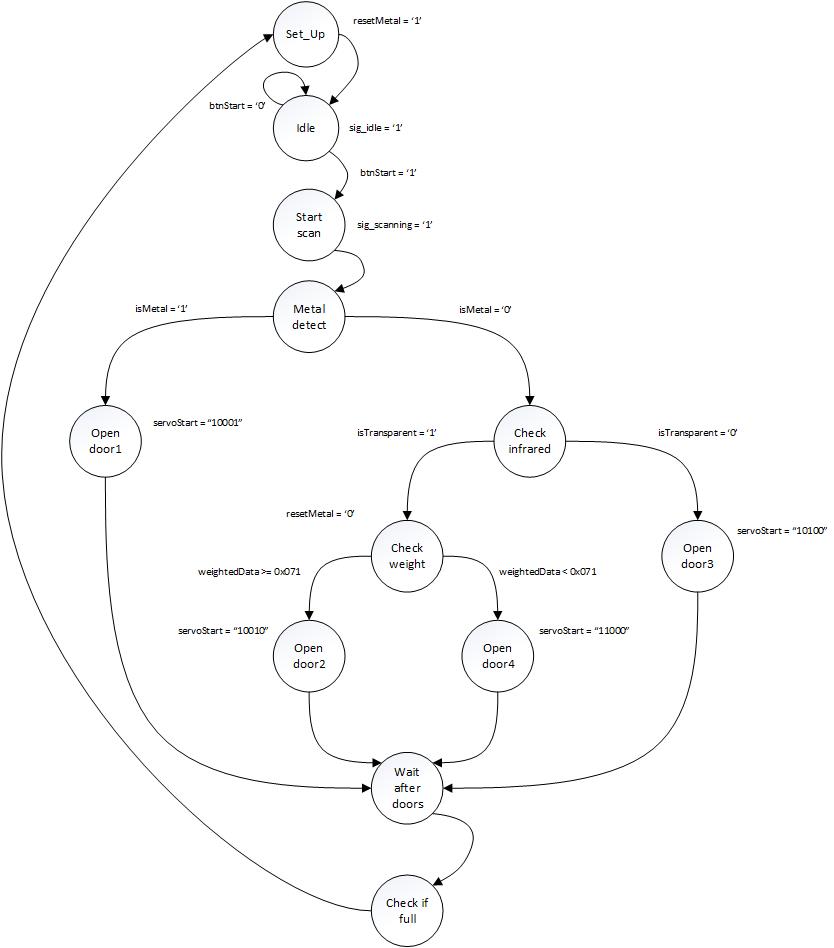
The servo\_status selects which servo should move. A configuration such as “10010” represents that the 5th servo and the 2nd servo should move.

|  |  |
| --- | --- |
|  | In the default\_state the FSM moves to the next state once the servo\_status input signal is changed. No motors move.  In the move\_servo state, the selected servos receive move signals different than “00”.  In practice, move\_servo5 can never be “00” in this state, since it is the trap door.  In this state, the doors open, letting the object fall. The doors stay open for 5 seconds.  In the move\_back state, each servo motor goes back at its initial position. It takes approximately 2 seconds for the motors to get back in the initial state. |

## Main



|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Direction** | **Description** |
| CLK | 1 | Input | 125MHz provided by the ZYBO board |
| btn | 4 | Input | the four buttons of the ZYBO board |
| echo | 4 | Input | signal received by the board after the trigger signal was sent |
| DOUT | 1 | Input | clock signal for the hx711 |
| isMetal | 1 | Input | signal received from the Arduino Nano |
| isTransparent | 1 | Input | signal received from the Arduino Nano |
| trig | 4 | Out | trigger signal with 10 μs HIGH level that will be sent to the ultrasonic module |
| servo\_signals | 5 | Out | signals used to control the servo motors |
| led | 4 | Out | ZYBO leds |
| sck | 1 | Out | 500kHz signal driven to hx711 |
| resetMetal | 1 | Out | signal used to sample a new null value for the frequency of the metal detector |
| sig\_idle | 1 | Out | signal sent to the Arduino in order to display the idle message on the LCD |
| sig\_scanning | 1 | Out | signal sent to the Arduino in order to display the start of the scan message on the LCD |
| binFull | 4 | Out | signal sent to the Arduino in order to display which bin is full |



The FSM starts with the set\_up state where the metal detector is calibrated. The resetMetal is driven to the Arduino where the logic will sample a new null value as a reference for the oscillating circuit. The next state is idle, where we send the sig\_idle signal used for displaying messages on the LCD.

The next state comes only when the user presses the button and the system goes in the start\_scan state, if an object was introduced. There, a message is displayed stating that the scanning process has started, and the user is kindly asked to wait until the processing is finished before introducing another object.

In the next state, the system waits for an input from the metal detector, to determine if the object is made of metal or not. If it’s a metallic object, the next state will open the corresponding door.

If it is not a metal, then the state checkInfrared tests whether the object is transparent or not. If not, the object is regarded as waste and the corresponding door will open. If the item is transparent, the state check weight checks the weight of the item and depending on the threshold (0x071 which we calculated that it represents 100 grams), the object is placed in the corresponding compartment.

The open\_door states are similar and their purpose is the same, to open a certain door that leads to a compartment. In those states, we assert the servoStart signal to open the trapdoor of the processing room and depending on which door should open, that particular bit is set.

In the wait\_after\_doors state we wait approximately 9 seconds for the doors to properly terminate their movement and for the object to reach the corresponding compartment.

After that, we check if any of the bins is full in the check\_if\_full state and we remain there until the bin is emptied. A corresponding message is displayed.

In the end, the system goes back in the set\_up state in order to calibrate the metal detector and then, finally, in the idle state, waiting for another input.

# 

# 

# Discussion

## Problems Encountered

One of the problems encountered throughout the development of the project was the fact that the Zybo board works on 3.3V logic level, while most of the sensors used in the project communicate and function on 5V signals. After thorough research a solution was found: the bidirectional logic level shifter which can convert signals from a higher voltage to a lower one and vice versa, so we have converted all the 3.3V signals from the Zybo board into 5V signals that went to the sensors, and all the 5V signals from the sensors and from the Arduino boards to 3.3V signals that were connected to the Zybo board.

Another problem encountered was that the faces of the pyramid are made of plexiglass and the servos, which are connected to those faces, are not strong enough to remain steady if a heavy object falls on a pyramid face. This is why, when a heavy object falls from the scanning room, the other three doors instead of directing the item in the right compartment will be move by the force generated by the falling object and will possibly make the item to end in the wrong compartment. The solution we found to this problem was to add the constraint of introducing only objects that have a maximum weight of 0.5 kg.

Last but not least, because the project required the use of many sensors, this aspect implied that many wires would be used and our project was at a certain moment in a state when it looked rather messy, until we have grouped and bound all the wires corresponding to each sensor. This action lead to easier debugging and made the project to look nice rather than messy.

## Marketability

The project started from the following idea: build a machine which can solve one of humanity’s actual problems. After many brainstormings and hours of deliberating we have found a problem which we thought we could partially solve using our technological knowledge, i.e., making recycling easier and more common in everyday places like residences/homes and offices. However, because this is an actual problem of our society, we estimate that the system designed by us will not remain just a prototype, but will be implemented on a larger scale and will evolve to be part of tomorrow's technologies. For now, it’s use is clear in small locations, but it can be updated and restructured to be used in the industrial area too.

## Community Feedback

From our discussions with various people, we received only positive feedback for the idea and in particular from the local scout centre in our town. Currently we are in talks with the scouts to display our project at their centre, since we might find people willing to help us take this project to the next step.

We have also posted our video presentation on facebook and youtube. So far, we have gotten mostly positive feedbacks regarding the project. The only problem we had is due to the constraints pertaining to the objects that can be inserted in the trash bin. However, this is to be understood, since the trash bin is just a prototype.

# References

1. <http://dzlsevilgeniuslair.blogspot.ro/2013/07/diy-arduino-based-metal-detector.html>
2. <https://www.allaboutcircuits.com/projects/metal-detector-with-arduino/>
3. <http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf>
4. <https://www.xilinx.com/support/documentation/university/XUP%20Boards/XUPZYBO/documentation/ZYBO_RM_B_V6.pdf>
5. <http://users.utcluj.ro/~negrum/src/html/ca.html>
6. <https://learn.sparkfun.com/tutorials/getting-started-with-load-cells>
7. <https://www.jameco.com/Jameco/workshop/howitworks/how-servo-motors-work.html>

# Appendix A

The source code and the bit file are in the 22CJRO\_Project zip file.