First project Fundamentos de Arquitectura de Computadores: Combinational Logic, door with a password

1st Daniel Serrano
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
dsecan254@estudiantec.cr

***Abstract*—This paper presents the design and implementation of a digital circuit for opening and closing a door based on a binary password input. The system uses a piezoelectric sensor to detect knocks, each representing a binary digit, which are then serially passed to a shift register (74HC595) that outputs it in parallel. The resulting bit sequence is processed as a password that determines whether the access request is valid. When a correct password is entered, the circuit generates the necessary output signals to indicate the action of opening or closing the door. A state sensor is also incorporated to ensure that the action is only attempted if the door is in the appropriate position, so it is avoided that the door is opened or closed when it's already on the attempted state. Although the DC motor was not actuated in practice due to current limitations of the Arduino power supply and the absence of a transistor-based driver circuit, the project successfully demonstrates the design and simulation of the password recognition and control logic. This provides a solid foundation for future hardware improvements and a later complete and functional implementation.**

***Index Terms*—circuit, digital, logic gates, piezoelectric, Arduino, latch SR, decoder, shift register, DC motor, seven segment display**

I. Introduction

This paper documents the design and implementation of a digital circuit that detects a binary input generated by the presence or absence of a knock, using a clock signal and an 8-bit shift register (74HC595) [1]. A piezoelectric sensor is employed to capture the knocks, which are interpreted as binary bits and compared against a predefined password with decoders. When the correct sequence is entered, the circuit generates the signals required to open or close a door.

The remainder of this paper is organized as follows. The Developed Algorithms section describes the algorithm developed for processing the knock-based binary input and controlling the outputs. The Results section presents the experimental results, highlighting the functional requirements that were met or partially achieved. Conclusions section provides the conclusions of this work, while the Recommendations section outlines recommendations for future improvements. Finally, the references used throughout the development of this project are listed at the end of the paper.

II. Developed Algorithms

A. Input sensor

For the input sensor, it was decided to use a piezoelectric buzzer. This is "a device that can convert various physical forces, including pressure, vibration and temperature, into electrical charges that can be measured." [2]. How this accomplish it's function, is because this device contains a crystal, that when a force is applied to it, generates an electric charge. [2].

For the design, a diagram from Arduino webpage [3] was extracted, shown in figure 1.

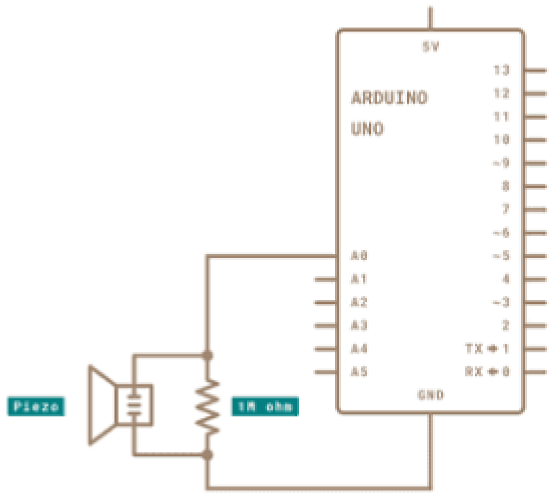

Fig. 1. Piezoelectric circuit diagram extracted from [3].

As seen on figure 1, the positive pin of the piezoelectric sensor is connected to pin A0 of the Arduino, and negative cable is connected to GND (ground). In the example, they use a 1M Ω resistor, however, in the case of the project, a 2k Ω resistor was used because these components were the ones available to avoid spending more than necessary and it still works fine.

Since the piezo generates a very small signal, it needs to be amplified. To prevent doing more work than necessary, the Arduino was used as an amplifier, so when in A0 it detects a pulse from the piezo, it sends a 1 from one of the digital pins, for this project, pin 7 was chosen as the output signal.
For this, the following Arduino code was made assisted by ChatGPT and modified slightly so the threshold coded to avoid noise, fits the conditions of the knock that should trigger the sensor.

```
const int piezoPin = 8;
const int piezoPinIn = A0;
const int tresholdPiezo = 40;
void setup(){
  pinMode(piezoPin, OUTPUT);
}
void loop(){
int sensorValue = analogRead(piezoPinIn);
  if (sensorValue > tresholdPiezo) {
    digitalWrite(piezoPin, HIGH);
  } else {
    digitalWrite(piezoPin, LOW);
  }
}
```

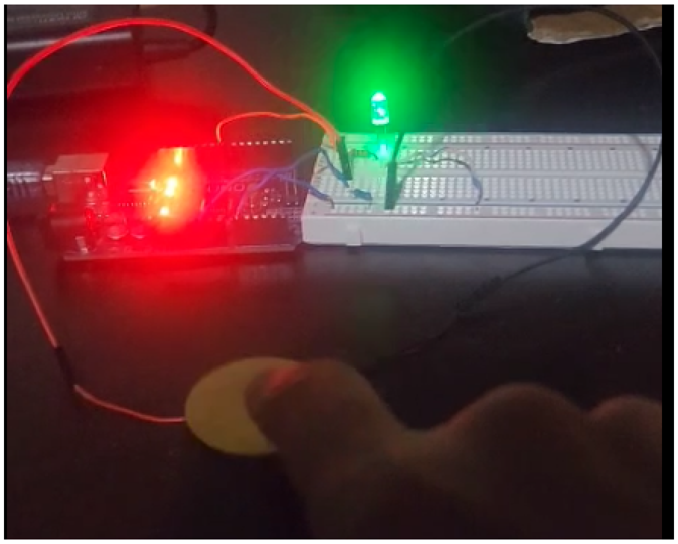

Fig. 2. Physically built sensor circuit with a piezoelectric sensor turning on a led.

On figure 2 it can be seen the circuit built physically and working to detect knock on the sensor.

B. 8 bit shift register (74hc595)

For managing the 8 bits of the input, a 74hc595 shift register was used. This integrated circuit is composed of 8 flip flops D connected in cascade [1], with a clock connected on pin 11 and 12 to allow the bits to move through each output. The clock was coded in the arduino with the following code

```
const int clockPin = 2; //salida del reloj
int clockState = HIGH;
int clockTime = 0;
void setup() {
  pinMode(clockPin, OUTPUT);

  Serial.begin(9600); // initialize serial
      monitor
}

void loop() {

  // CLOCK, DONT TOUCH
  digitalWrite(clockPin, clockState);
  if (clockTime >= 100) {
    clockTime = 0;
    clockState = !clockState;

    // Debug clock
    Serial.print("Clock␣toggled,␣state:␣");
    Serial.println(clockState);
  }
  delay(1);  // 1 ms
  clockTime += 1;
  // END CLOCK-------------
}
```

The clock was coded so every time clockTime variable reached 100, it meant 100ms has passed and the loop is delayed 1ms each time. This logic was thought so a single Arduino could control other components without delaying their function. The period T of the clock ended up being 200ms because it felt as enough time to knock normally, not too fast and not too slow.
On pin 13, a cable was connected to ground to enable the outputs of the flip flops by giving VCC to the not gates shown on page 7 of [1].
On pin 14, the serial input was connected, always receiving a logic 0 unless a pulse is sent. Because of the clock on pins 12 and 11, the 0 is always moving through the 8 outputs, which are located on pin 15 and 1 through 7 of the 74hc595 [1].
To hold the input, just the pulse is not enough, since the voltage for the logic 1 needs to be on pin 14 when the clock makes the shift. For this issue, a SR latch, using a NAND gate (74LS00) [4].

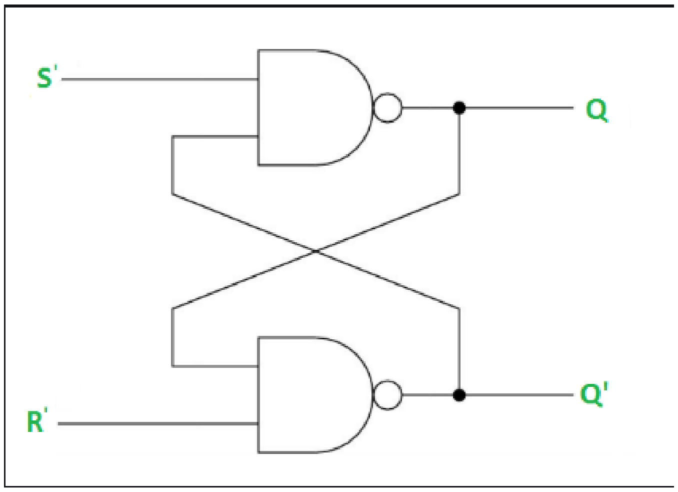

Fig. 3. SR latch using nand gates taken from [5]

TABLE I
SR LATCH VALUES

S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	X	X

The SR latch on figur 3 will hold the voltage on Q when a logic 1 is entered on the Set pin. And the reset will be connected to the first output of the 74hc595, so when the bit is already in the circuit flow, the latch resets and waits for a new set, staying with a 0. Following table I, where S is connected to Arduino pin 7 and reset to the 74hc595 pin 15. It starts with S=0 and R=0. When a pulse is sent and S=1 for less than a second, the S=1 and R=0 row happens, so Q sets to a 1. When both S and R go back to 0, Q holds the previous value, this would be connected to pin 14 of the 74hc595. When the clock shifts the bits on the flip flops D, through pin 15 of this shift register, a 1 is sent, so R=1 and S=0, therefore Q stores a 0, and since S=0 unless the sensor detects a knock, Q will remain on state 0 untill S=1 again.

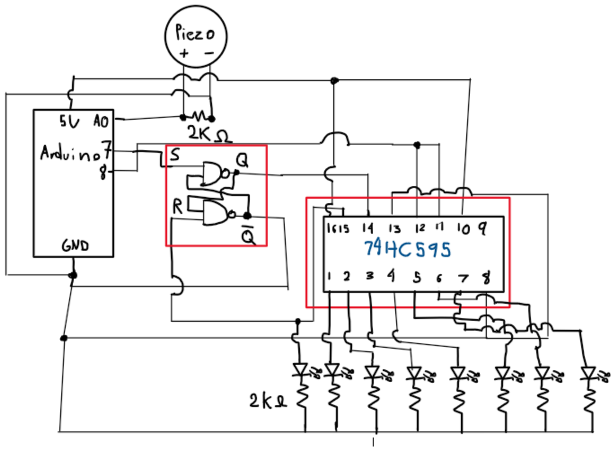

Fig. 4. Sensor with arduino and 74hc595 diagram

Figure 4 shows how the decoder and the 74hc595 connect according to what has been described in this subsection. On each output of the shift register, a LED to keep track of which outputs have a bit or not. Each LED cathode is connected to ground with a 2k Ω resistor to prevent damaging it with high currents. $\overline{Q}$ is connected to ground only, because it is not needed.

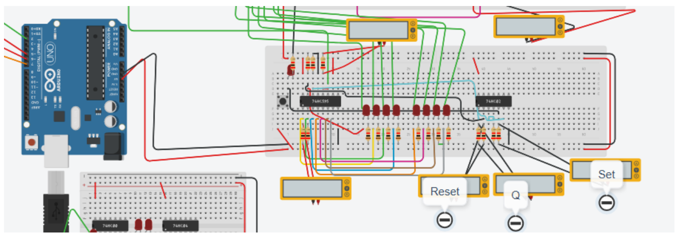

Fig. 5. Circuit built on Tinkercad software for testing

On figure 5 the circuit was built on the Tinkercad software for testing the logic planned.

C. Decoders

For the decoders, two passwords were randomly created of 8 bits:

- For opening: 10011101
- For closing: 11100111

To be able to detect this, the minterm for each password was calculated.

TABLE II
OPEN PASSWORD TABLE.

A	B	C	D	E	F	G	H	Open
1	0	0	1	1	1	0	1	1

TABLE III
CLOSE PASSWORD.

A	B	C	D	E	F	G	H	Close
1	1	1	0	0	1	1	1	1

Knowing which inputs are 1 and 0, and knowing this is the only combination in each decoder that will output a 1, the minterm for each can be easily calculated as follows.

$$Open = A\,\overline{B}\,\overline{C}\,D\,E\,F\,\overline{G}\,H \tag{1}$$

$$Close = A\,B\,C\,\overline{D}\,\overline{E}\,F\,G\,H \tag{2}$$

With the calculated equations (1) and (2), the circuit can be designed. This equations can't be simplified further than just combining some negations into one input of the AND gate because for example, in the Open minterm case, BC are both negated, so they can share the NOT gate that enters the AND gate. This is done also on the close minterm with D and E, this is not necessarily a boolean simplification but a wiring simplification that works and saves resources.

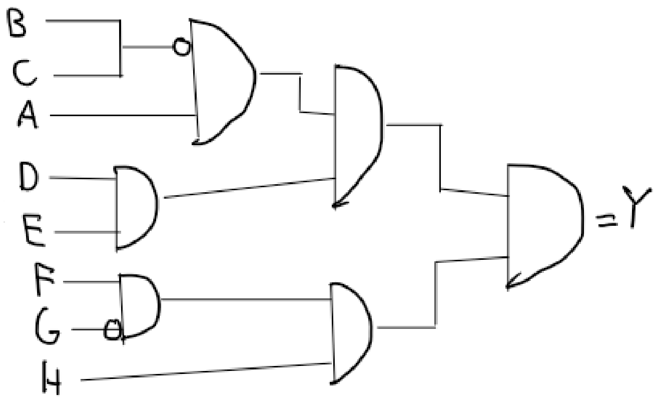

Fig. 6. Open password decoder circuit diagram.

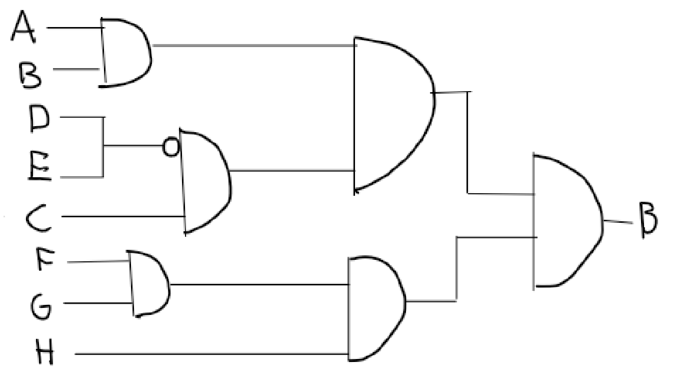

Fig. 7. Close password decoder circuit diagram.

On the figures 6 and 7 the minterm of equations (1) and (2) can be represented as circuits using AND gates. This was first tested on the Tinkercad software for simulation as shown on figure 8 and then built physically as shown on figure 9.

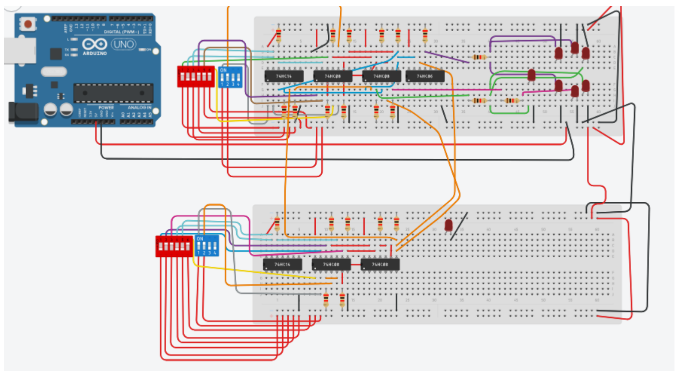

Fig. 8. Decoders on Tinkercad simulation software.

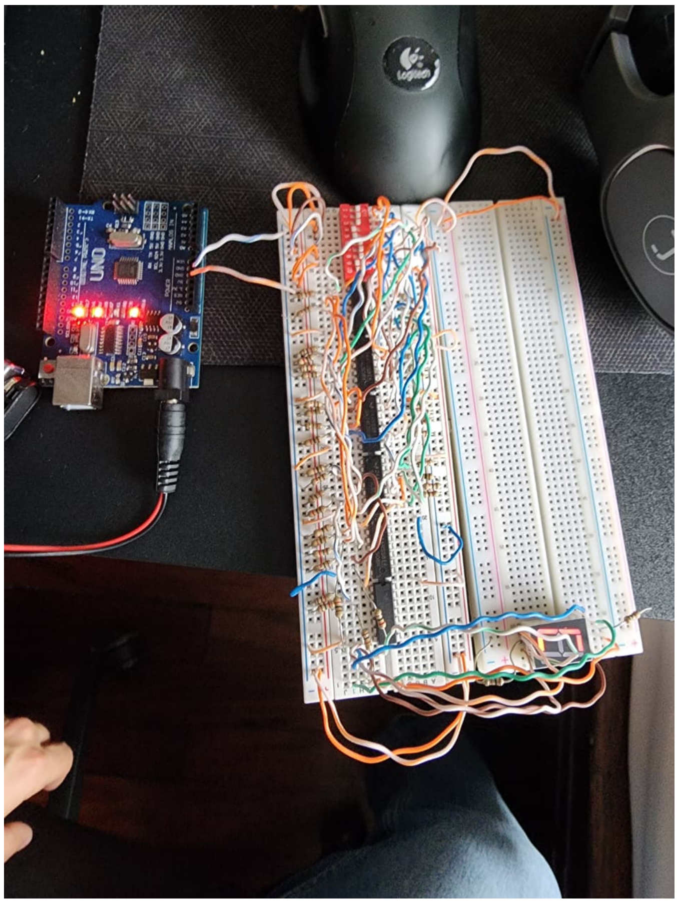

Fig. 9. Physically implemented decoders.

On the physical implementation, 74ls08N AND gates [6] were used. For the inverters, the SN74LS14N [7] where used. For these gates, different from the hc variant, each input needed a resistor to ground in parallel, so 2k Ω resistors were used. On Tinkercad, the same resistors were used but in the output pins of the AND gates. The circuit on figure 9 combines both decoders and inputs designed on 8.

D. 7 segment display combinatory circuit

For lighting the correct segments of the 7 segment display, a combinatory circuit was made, where a table was first made to understand which segments need to be lit up when which signal is sent by each decoder; this analysis can be shown on table IV.

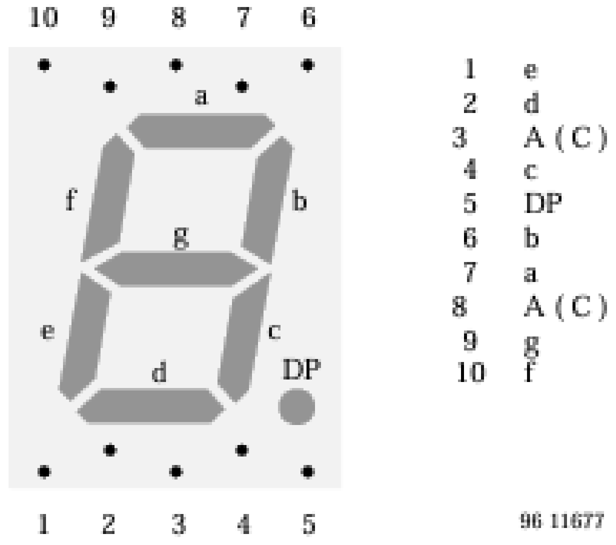

Fig. 10. Diagram of a 7 segment display, extracted from [8]

TABLE IV
ANÁLISIS DE LOS 7 SEGMENTOS DESEADOS A ENCENDER.

A	B	a	b	c	d	e	f	g
1	0	1	1	1	0	1	1	1
0	1	1	0	0	1	1	1	0

For the analysis on table IV, pin 5 is not taken into consideration. This table shows that for an A of Open in spanish is shown, segments a, b, f, g, e and c should light up. For a C of closed, segments a, f, e and d should light up. When analyzing, a similarity can be noticed between segment a, e and f, when A and B decoders have diferent values from each other, these segments are always on, this can be compared with the XOR table values, as shown in table VII, only when A and B are different from each other, it outputs a 1, so this is the first step, a XOR between A and B can be connected to segments a, e and f. Next, segments b, c and g share the value on both conditions of interest, only when A has a value of 1 and B a 0, these segments light on, so the minterm for these can be calculated, like on the table V, giving the equation (3). Finally, for the last segment, d, the minterm is obtained from table VI giving the equation (4).

TABLE V
SEGMENTS B, C AND G MINTERMS

A	B	Sal	Mintermino
1	0	1	$A\overline{B}$
0	1	0	-

TABLE VI
SEGMENT D MINTERM

A	B	Sal	Mintermino
1	0	0	-
0	1	1	$B\overline{A}$

$$b, c, g = A\overline{B} \tag{3}$$

$$b, c, g = B\overline{A} \tag{4}$$

TABLE VII
XOR TRUTH TABLE

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Finally, with these equations, the circuit for lightning the segments of the 7 segment display can be designed as in the diagram of figure 11. This circuit can be seen simulated on figure 8, in this case, LEDs where used to simulate the seven segment display because the one included in the software has the internal diodes inverted. The physical implementation can be seen on figure 9. For the AND gates, the 74ls08N [6] where also used. For the XOR gate, the 74LS86N [9] was used, which as every LS gate, has a resistor connected to each input, in this case, a 2K Ω resistor. For the NOT gate, the SN74LS14N [7] is also used.

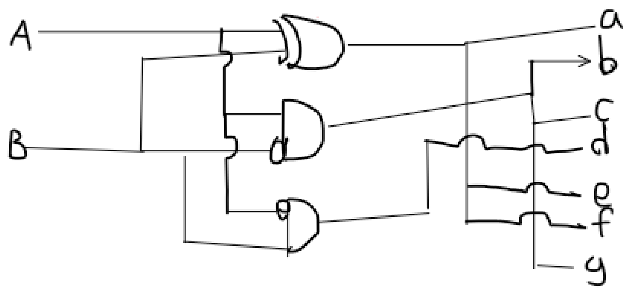

Fig. 11. Combinatory circuit diagram for the 7 segment display

.

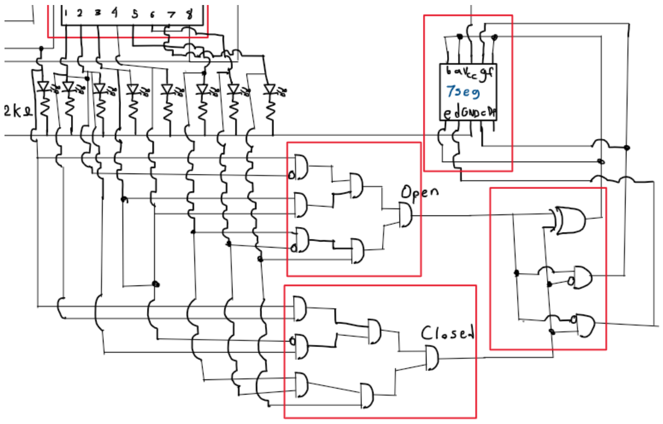

Fig. 12. Diagram of the decoders attached

.

E. Motor decoupling

A motor has two cables, one positive and one negative. Depending on the direction of the flow of the current, the motor will spin in the direction of the current. Based on knowing this, the outputs of both decoders can be connected on both sides of the motor, each cable with a 2k Ω to prevent damaging the motor for intense currents.
Now, considering that from the shift register, each bit is constantly moving each 100ms, the pulse from the open or close gate will appear only for 100ms. To keep the motor running until the door open or closes, even if the pulse is gone, a latch SR with AND and NOR gates was implemented. The NOR version was added because an SR latch can be done the same way as with a NAND gate according to figure 13 [5], and there were no more NAND gates available, so to prevent spending more money, one of the latches was made with a NOR gate, physically with the TC74HC02AP [7]. Analyzing again table I, with the pulse of the output of the decoder to open the door, one SR latch is set to 1, and another latch, the one with the NOR gates, is set to 1 with the pulse of the decoder to close the door.

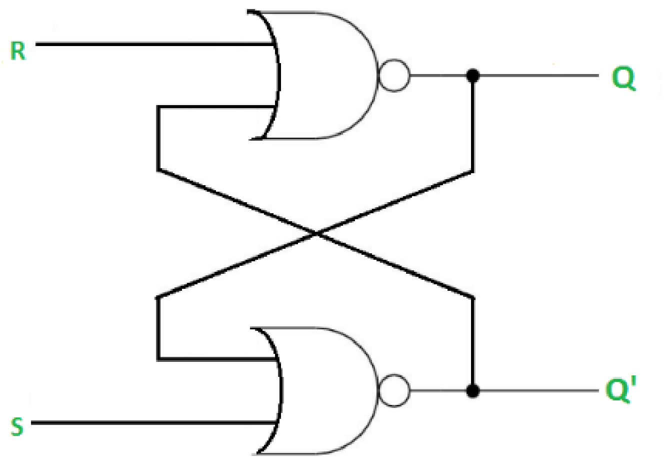

Fig. 13. Latch SR with NOR gates, extracted from [5]

.

But what happens when the door reaches the desired position? How will it be reset? For stopping the motor and resetting the latches, a Ultrasonic Sensor was added, specifically the HCSR04 [10]. This was implemented as the stopping sensor because it is what was at hand to save money. The logic of the sensor was coded in the Arduino with the following code

```
const int trigPin = 6;     // TRIG del HC-SR04
const int echoPin = 5;     // ECHO del HC-SR04
const int outputPin = 7;   // digital output

long duration;
int distance;
const int threshold = 200; // cm distance to
    change state

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(outputPin, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  // Generate triger pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure response time
  duration = pulseIn(echoPin, HIGH, 30000);

  if (duration == 0) {
    distance = 999; // out of range
  } else {
    distance = duration * 0.034 / 2; //
        convert to cm
  }

  //Digital signal, HIGH if out of range, LOW
      if in range
  if (distance >= threshold) {
    digitalWrite(outputPin, HIGH);
  } else {
    digitalWrite(outputPin, LOW);
  }
}
```

The sensor is coded so, when the door is out of range, it would be open and the sensor outputs a logic 1. When it is in a range of 2cm or less, the door is closed so the sensor outputs a logic 0. The code was generated with the help of ChatGPT, which was manually modified for the desired distance and to trigger the state switch and commented for better understanding.
Now, this output is connected to another latch SR with NAND gates, the output will be connected to the set pin of this latch and inverted to the reset pin. The goal of this setup comes from the latch truth table shown in table I. Depending if S=0

and R=1, or viceversa, the Q and $\overline{Q}$ are switched as well. Tinkering with this behavior, the Q is connected to the reset of the latch that holds the close pulse, so while S=0, meaning while the door is open or still closing, the closing latch will not reset, but as soon as the 2cm distance is reached, S will change to 1 and the closing latch will have a 1 in the reset and therefore will stop the motor. $\overline{Q}$ will be connected to the latch that will hold the pulse for opening the door, so as long as S=1, $\overline{Q}$ will be 0 and the reset on this latch will be 0 also, but as soon as the door is out of range, S will become a 1 and the R of the opening latch will become 1, stopping the motor from opening further the door. On figure 14, this implementation can be clearly seen, on the top, from right to left, a NAND and NOR gates are placed to build the mentioned SR latches. This can be seen on the physical implementation on figure 15.

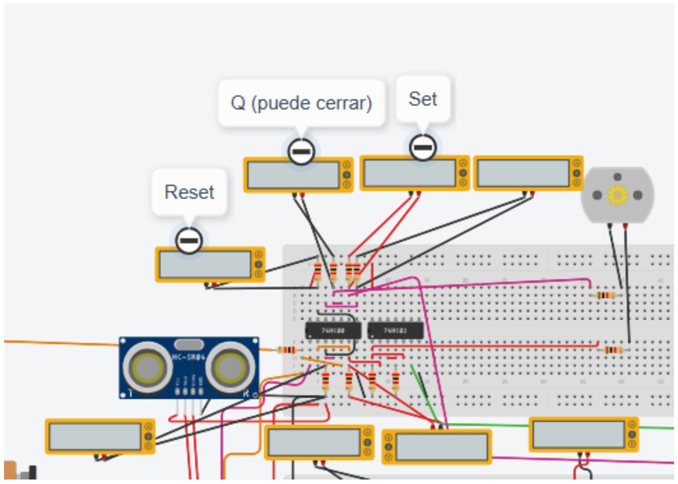

Fig. 14. Decoupling of the decoders to the motor with ultrasonic sensor, simulated on Tinkercad

.

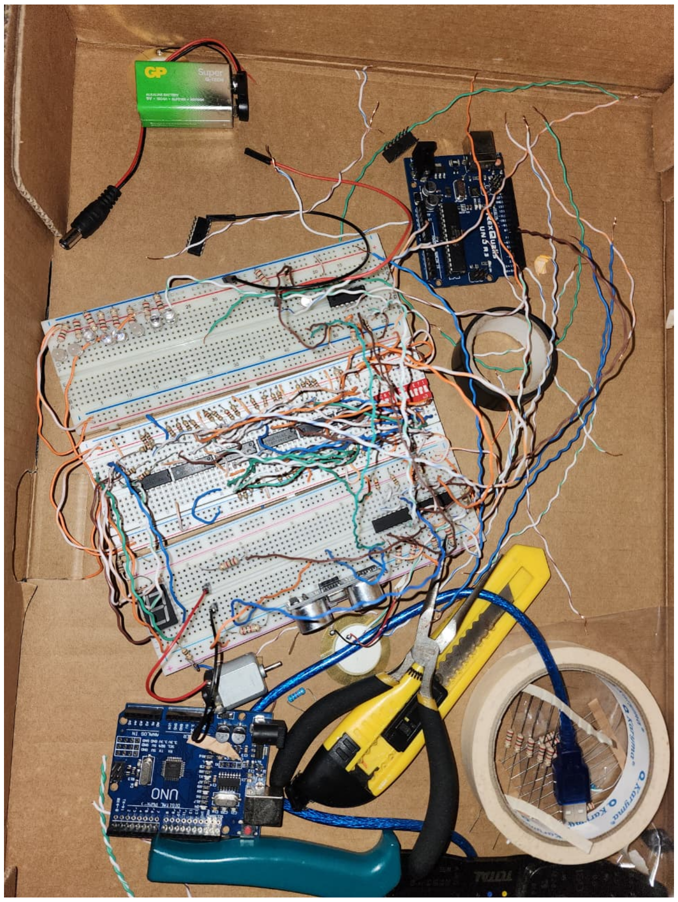

Fig. 15. Final circuit, with first part disconnected post project defense

.

F. Final Arduino Code

The final Arduino program was developed to perform three main tasks: amplifying the sensor pulse, generating the clock signal, and controlling the ultrasonic sensor. The complete implementation is presented below.

```
const int clockPin = 2;
const int piezoPin = 8;
const int piezoPinIn = A0;
int clockState = HIGH;
int clockTime = 0;
const int tresholdPiezo = 40;

const int trigPin = 6;     // TRIG del HC-SR04
const int echoPin = 5;     // ECHO del HC-SR04
const int outputPin = 7;   // digital sensor
    out

long duration;
int distance;
const int threshold = 200; // distance in cm
    to activate HIGH

void setup() {
  pinMode(clockPin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(outputPin, OUTPUT);
  pinMode(piezoPin, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  // Generate trigger pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure response time
  duration = pulseIn(echoPin, HIGH, 30000); //
       timeout 30 ms

  if (duration == 0) {
    distance = 999; // out of range
  } else {
    distance = duration * 0.034 / 2; //
        convert to cm
  }

  //Digital signal, HIGH if out of range, LOW
      if in range
  if (distance >= threshold) {
    digitalWrite(outputPin, HIGH);
  } else {
    digitalWrite(outputPin, LOW);
  }

  // Piezo
  int sensorValue = analogRead(piezoPinIn);
  if (sensorValue > tresholdPiezo) {
    digitalWrite(piezoPin, HIGH);
  } else {
    digitalWrite(piezoPin, LOW);
  }

  // Debug piezo
  //Serial.print("Piezo analog value: ");
  //Serial.print(sensorValue);
  //Serial.print(" | Piezo output: ");
  //Serial.println(digitalRead(piezoPin));

  // CLOCK, DONT TOUCH
  digitalWrite(clockPin, clockState);
  if (clockTime >= 100) {
    clockTime = 0;
    clockState = !clockState;

    // Debug clock
    Serial.print("Clock␣toggled,␣state:␣");
    Serial.println(clockState);
  }
  delay(1);  // 1 ms
  clockTime += 1;
  // END CLOCK-------------
}
```

III. Results

The results of the project are presented in three parts: the input sensor and serializer, the decoders, and the motor control.

A. Input Sensor and Serializer

The sensor behaved as expected. A LED was connected in parallel to the output of Arduino pin 7, and each knock made it light up briefly, confirming a successful detection of the knock. However, the serializer did not work correctly due to a design flaw. The latch designed to hold and reset bits was insufficient, since the reset also triggered the loading of the next bit. As described in the 74HC595 datasheet [1], the shift register is implemented with a two-stage flip-flop design (p. 7). This caused the output to generate two consecutive pulses instead of the intended sequence, preventing the correct transmission of the password.

B. Decoders

The decoder stage worked as expected. When tested with eight switches as inputs, the closing password produced the expected output on the seven segment display, as shown in figure 9. For the opening password, the display also behaved correctly; however, occasional flickering of the segments *a*, *f*, *e*, and *g* was observed. This behavior was likely caused by unstable connections or loose resistors, and it could not be corrected in the available time.

C. Motor Control

The motor control stage worked only in simulation but failed in the physical implementation. In simulation, the motor responded correctly when activated. Physically, however, the motor did not spin. This was likely due to the Arduino's limited current capacity, which was further divided among other components. The design required a transistor to act as a switch, enabling an external power source (e.g., a 9V battery) to supply the necessary current for the DC motor. Since this was not implemented, the motor remained inactive despite a measurable voltage of approximately 2V across its terminals.

Conclusions

The development of this project demonstrated that with a solid understanding of electronic components and a modular approach, complex designs can be effectively carried out by a single person. Nonetheless, collaboration can be valuable to identify design flaws that may be overlooked due to human error, as occurred in this work. Despite these challenges, approximately 60% of the project's functionality was achieved. The missing elements were the addition of a transistor and second power source to properly drive the DC motor, and the sequential circuit required to reset the latch that holds the sensor pulse. On the other hand, the decoder implementation was successful, consistently producing the correct output when the input matched the predefined password. These results confirm the utilty of the design made, while also highlighting areas for improvement in a future improvement of this project or rework.

Recomendations

For a project like this, it is highly recommended to plan beforehand as modularly as possible. It is of great help to build each part step by step and test each one individually. Ask yourself what the inputs are and what this section will output; once each section outputs what is expected, it is guaranteed it will work.

It is also recommended to test the components before incorporating them to the circuit, as some of them could come faulty or misinterpreted when building. Doing this will prevent having to separate the component from the rest of the circuit and losing time testing it after.

Consider adding a second power source for motor current. The Arduino does not provide enough current for a DC motor to work properly, so adding a transistor as a switch that can be activated to feed a DC motor is a great choice to be able to activate it properly.

References

[1] *74HC595 : 8-Bit Serial-Input/Serial or Parallel-Output Shift Register with Latched 3-State Outputs ON Semiconductor*, ON Semiconductor, 2000. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/pdf/12198/ONSEMI/74HC595.html

[2] L. APC International. (2024) How do piezoelectric sensors work? Accessed: Sep. 18, 2025. [Online]. Available: https://www.americanpiezo.com/blog/how-piezoelectric-sensors-work/

[3] ARDUINODOCS. (2024) Detect a knock. Accessed: Sep. 16, 2025. [Online]. Available: https://docs.arduino.cc/built-in-examples/sensors/Knock/

[4] *SN74LS00 Quad 2-Input NAND Gate*, ON Semiconductor, 1999. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/download/12609/ONSEMI/74LS00N.html

[5] G. for Geeks. (2025) Latches in digital logic. Accessed: Sep. 22, 2025. [Online]. Available: https://www.geeksforgeeks.org/digital-logic/latches-in-digital-logic/

[6] *SN74LS08N Quad 2-Input AND Gate*, ON Semiconductor, 1999. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/download/12619/ONSEMI/74LS08.html

[7] *SN74LS14 Schmitt Triggers Dual Gate/Hex Inverter*, ON Semiconductor, 2001. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/download/106044/ONSEMI/SN74LS14N.html

[8] *7mm Seven Segment Display*, Vishay Telefunken, 1999. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/download/26428/VISHAY/TDSR115.html

[9] *SN74LS86 Quad 2-Input Exclusive OR Gate*, ON Semiconductor, 1999. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/download/12663/ONSEMI/SN74LS86N.html

[10] *HCSR04 Ultrasonic Sensor*, Elijah J. Morgan, 2014. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/download/1132204/ETC2/HCSR04.html