Wrocław University of Science and Technology

MICROCONTROLLERS PROJECT

Faculty of Electronics, Photonics and Microsystems

Theme of class: Final Report

Students:
1. Stanislav Kustov 275512

Date of class: 2025-02-04

Submission Date: 2025-02-04

Lab assistant: Andrzej Grobelny

Contents

1 Introduction **2**
1.1 Purpose and Applications . . . 2
1.2 How It Works . . . 2
1.3 Technologies Used . . . 3
1.4 Overview . . . 3

2 Assumptions **4**
2.1 Functional Assumptions . . . 4
2.2 Design Assumptions . . . 4
2.3 Hardware Components . . . 4
2.4 Software Assumptions . . . 4

3 System Design and Electrical Schematic **5**
3.1 Programming Interface . . . 5
3.2 Power Supply and Battery . . . 6
3.3 LED Configuration and Resistors . . . 6
3.4 Capacitors for Filtering . . . 6
3.5 Mode Selection and Button Interface . . . 6
3.6 Current Consumption and Power Considerations . . . 6

4 PCB **7**
4.1 PCB Layout and Structure . . . 7
4.2 Programming Interface . . . 7
4.3 Component Placement and Routing . . . 8
4.4 Manufacturing and Assembly . . . 9
4.5 Power Supply and Circuit Protection . . . 10
4.6 User Interaction . . . 11
4.7 Challenges Faced During Development . . . 11
4.7.1 Component Selection and Availability . . . 11
4.7.2 PB7 Port Configuration Issue . . . 11
4.8 Different LEDs Modes . . . 12
4.8.1 LED Pin Definitions . . . 12
4.8.2 Button Handling . . . 12
4.8.3 Turning LEDs On and Off . . . 13
4.8.4 Setup Function . . . 13
4.8.5 Loop Function and Button Logic . . . 14
4.8.6 LED Modes . . . 14
4.8.7 Heart Animation . . . 15

5 Conclusion **17**

1 Introduction

The **Heart-Shaped LED PCB** is a custom-designed electronic project that blends **microcontroller programming, LED lighting effects, and PCB design** into a compact and visually appealing form. Originally, this project was meant as a **personalized gift**, which is why the PCB is designed in the shape of a **heart**. The goal was to create a meaningful and functional electronic device that not only looks unique but also showcases different lighting effects.

1.1 Purpose and Applications

This project was initially conceived as a **gift**, which influenced both the design and the functionality of the PCB. The **heart shape and LED pattern** were carefully chosen to emphasize its personal and decorative value. However, beyond being a unique present, this project also serves as an example of **custom PCB design, microcontroller programming, and hardware optimization**.

The PCB is based on an **ATmega328P microcontroller**, the same chip used in **Arduino** boards, making it easy to program and customize. It features **16 LEDs arranged in a heart shape**, which can be programmed to display different lighting effects such as:

- Steady glow
- Pulsating (breathing effect)
- Blinking patterns
- Dynamic animations

These effects create an engaging and visually striking appearance, making the project suitable not just as a gift, but also as an educational tool for learning about microcontrollers, PCB design, and LED control.

1.2 How It Works

The system consists of the following key components:

- **Microcontroller Unit (MCU):** An **ATmega328P-AU** is used to control the LED animations and handle user input.
- **LED Array:** 16 LEDs arranged in a heart shape, each controlled via the microcontroller.
- **Power Supply:** The device is powered by a **coin cell battery**, making it compact and portable.
- **Programming Interface:** To upload code to the microcontroller, an another **Arduino Uno was repurposed as a programmer**. This method allowed the design to remain minimalistic without the need for a dedicated USB interface.
- **Modes & Effects:** Different LED animations can be pre-programmed, offering a variety of lighting patterns.

1.3 Technologies Used

This project was developed using:

- **Altium Designer** For PCB design and layout.
- **Arduino IDE** For writing and uploading firmware to the microcontroller.
- **ATmega328P-AU** The core microcontroller handling the logic.
- **Surface-Mount LEDs & Components** To achieve a compact design with efficient power usage.

1.4 Overview

This report will cover:

- The hardware design process and PCB layout.
- Programming techniques used for controlling LEDs.
- The use of an Arduino Uno as a programmer.
- Challenges encountered and improvements for future versions.

By combining **electronics, design, and embedded programming**, this heart-shaped PCB serves as a beautiful, functional, and educational project that showcases creativity in hardware development.

2 Assumptions

2.1 Functional Assumptions

- The LED system will operate according to pre-programmed sequences.
- The device is designed to work with a single microcontroller without external communication modules.
- The power supply is assumed to be a coin cell battery, ensuring portability.
- The system will allow basic control over different LED modes.

2.2 Design Assumptions

- The PCB is designed to be compact and aesthetically pleasing.
- All components are surface-mounted to keep the form factor minimal.
- The heart-shaped layout is fixed and will not change dynamically.
- The design does not include an external interface for real-time user input beyond power on/off.

2.3 Hardware Components

- The microcontroller is an ATmega328P-AU with built-in LED control logic.
- The LED system consists of 16 individual surface-mounted LEDs.
- A coin cell battery is used as the primary power source.
- The PCB includes minimal external components to keep the design elegant and functional.

2.4 Software Assumptions

- The firmware is written in C/C++ using the Arduino IDE.
- The microcontroller is programmed using an Arduino Uno as an ISP programmer.
- The system does not require external sensors or communication interfaces.

3 System Design and Electrical Schematic

The heart-shaped PCB was designed with both aesthetics and functionality in mind. While the schematic may appear somewhat cluttered and unstructured, it effectively serves its purpose, ensuring that the microcontroller successfully drives the LEDs while maintaining power efficiency.

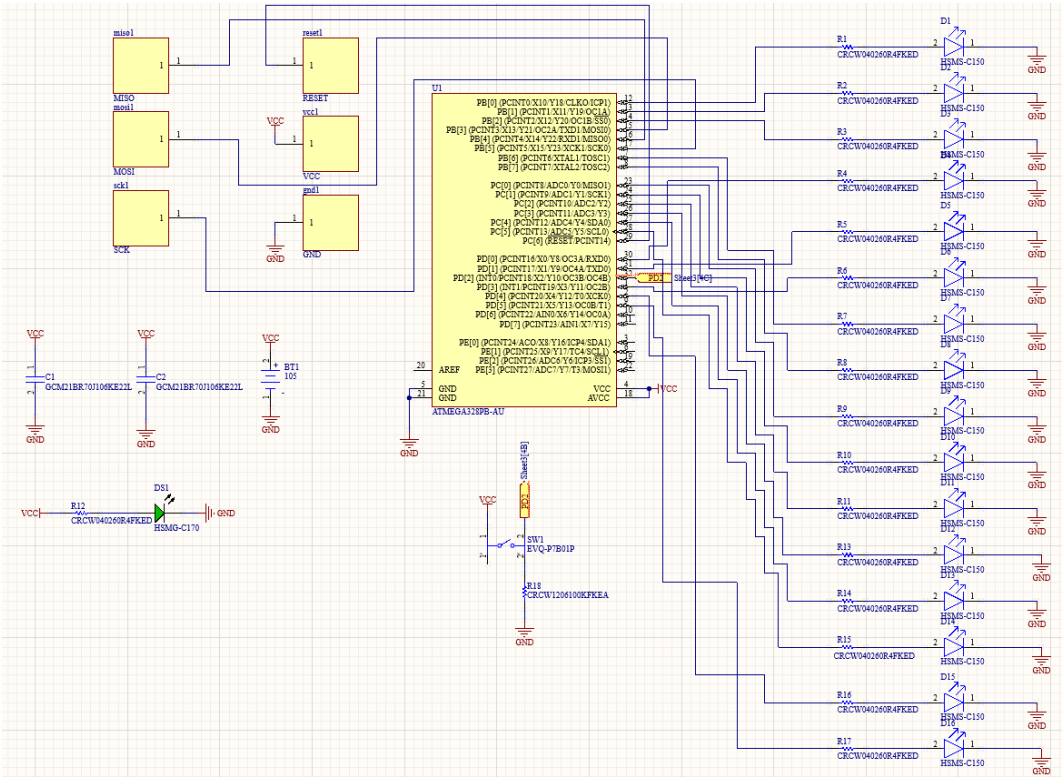


Figure 1: Electrical Schematic

3.1 Programming Interface

The blocks in the upper left corner of the schematic represent custom-designed pads for programming the microcontroller. These include:

- **MISO** Master In Slave Out
- **MOSI** Master Out Slave In
- **SCK** Serial Clock
- **RESET** Resets the microcontroller for programming
- **VCC** Power supply
- **GND** Ground

These programming pads allow the ATmega328P microcontroller to be flashed using an external programmer, ensuring the board remains minimalistic and free from additional connectors.

3.2 Power Supply and Battery

The system is powered by a **3.3V coin cell battery** with a capacity of **660mAh**. The choice of this power source ensures portability while providing sufficient energy to operate the LEDs and microcontroller efficiently. A **green LED** (model: **HSMS-C170**) is used to indicate power status.

3.3 LED Configuration and Resistors

The circuit features **16 red surface-mount LEDs** (model: **AVGO HSMS-C150**), which are connected to the digital pins of the microcontroller. Each LED is driven through a series resistor to limit the current and prevent damage.

From the LED datasheet, the forward voltage of each LED is approximately **1.9V**, and the typical forward current is **15mA**. Using Ohms Law, we calculate the appropriate resistor value:

$$R = \frac{V_{supply} - V_{forward}}{I_{LED}} = \frac{3.3V - 1.9V}{15mA} = 93.3\Omega \tag{1}$$

We use **Vishay CRCW040260R4FKED** resistors with a value of **91**, which is close to the calculated ideal resistance. These resistors are extremely small, with a package size of **0402** (**1.0mm 0.5mm**), which is about **twice the thickness of a human hair**. Their small size is crucial for maintaining the compact design of the PCB.

3.4 Capacitors for Filtering

To ensure stable operation of the microcontroller and LEDs, **ceramic capacitors** (model: **GCM21BR70J106KE2L**) are used for power supply filtering. These capacitors help reduce voltage fluctuations and noise in the circuit, ensuring reliable performance.

3.5 Mode Selection and Button Interface

The PCB also includes a small **tactile switch** (model: **ATK0000CE20**), which allows the user to switch between different LED lighting modes or turn the LEDs on/off. When pressed, this button pulls an input pin of the microcontroller low, triggering a change in state.

3.6 Current Consumption and Power Considerations

To estimate the total current draw, we assume all 16 LEDs are turned on simultaneously:

$$I_{total} = 16 \times 15mA = 240mA \tag{2}$$

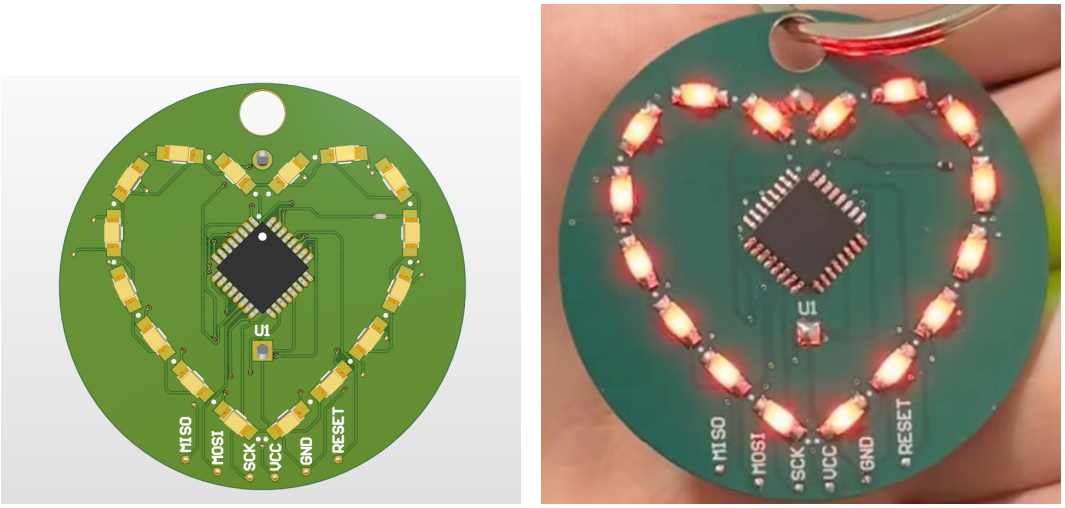
Given the battery capacity of **660mAh**, the estimated runtime can be calculated as:

$$t = \frac{660mAh}{240mA} \approx 2.75\,\text{hours} \tag{3}$$

This means the PCB can operate continuously for about **2.75 hours** on a full charge. However, since different lighting modes use fewer LEDs at any given time, actual battery life is likely longer.

4 PCB

The heart-shaped PCB is designed to combine both aesthetics and functional electronics. The design incorporates an ATmega328P microcontroller, LEDs, a programming interface, and a compact power supply to ensure seamless operation. The layout and component placement were carefully selected to maximize efficiency and usability.



(a) PCB Design (b) Real Life View

Figure 2: PCB Design and Real Life View of the heart-shaped PCB.

4.1 PCB Layout and Structure

The PCB features a **5 cm x 5 cm** circular layout with a heart-shaped LED arrangement. At the top of the PCB, there is a **large mounting hole**, which allows the board to be used as a keychain or attached to a lanyard. The **center of the PCB** is dedicated to the ATmega328P microcontroller, symbolizing the heart of the design.

All power supply components, capacitors, and resistors are placed on the **back side of the PCB**, beneath the battery holder. This ensures protection against accidental damage while maintaining a clean appearance on the front side. The **mode selection button** is also positioned on the back, near the edge, making it convenient for the user to press.

4.2 Programming Interface

At the bottom of the PCB, there are custom-made programming pads designed for in-system programming (ISP) of the microcontroller. These pads include:

- **MISO** Master In Slave Out
- **MOSI** Master Out Slave In
- **SCK** Serial Clock
- **RESET** Resets the microcontroller for programming

- **VCC** Power supply
- **GND** Ground

These pads allow for direct flashing of the ATmega328P without requiring additional headers, keeping the design minimalistic. An extbfArduino Uno was repurposed as a programmer, making it easy to upload firmware directly to the onboard microcontroller.

4.3 Component Placement and Routing

The PCB routing was carefully designed to optimize the signal integrity and minimize crosstalk. The **top layer** of the PCB consists of power and signal traces, while the **bottom layer** houses additional traces and passive components, such as resistors and capacitors.

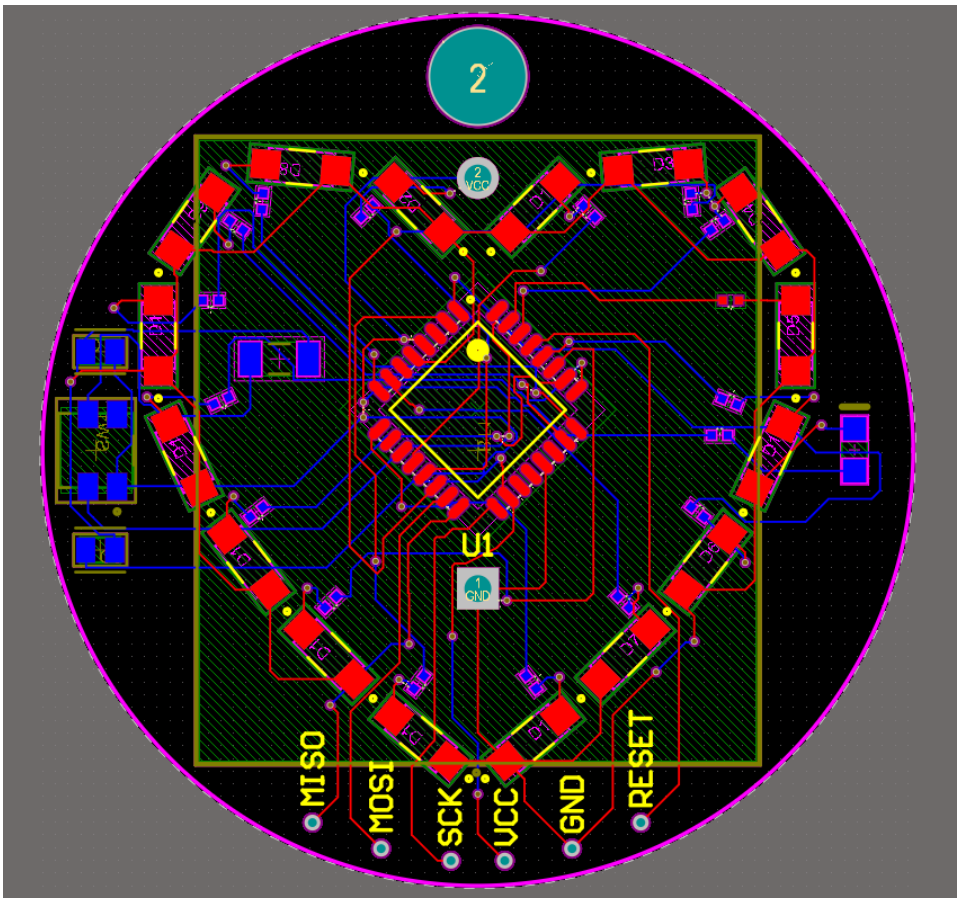


Figure 3: Component Placement

The traces for the LEDs follow a symmetric heart shape, ensuring a uniform and visually appealing layout. The **LEDs are arranged in a perimeter heart pattern**, with each LED connected to an individual GPIO pin of the microcontroller. The **current-limiting resistors** (0402-sized, **1.0 mm 0.5 mm**) are placed close to the LEDs to minimize trace resistance.

4.4 Manufacturing and Assembly

The PCB was manufactured in **China**, while all electronic components were ordered from **Mouser Electronics** in the United States. The entire assembly process was performed manually using **professional soldering equipment**. Given the **small size of components**, including the 0402-sized resistors, the soldering process required the use of a **microscope** to ensure accuracy. Each component was placed carefully by hand and soldered using precision techniques to achieve a reliable connection.

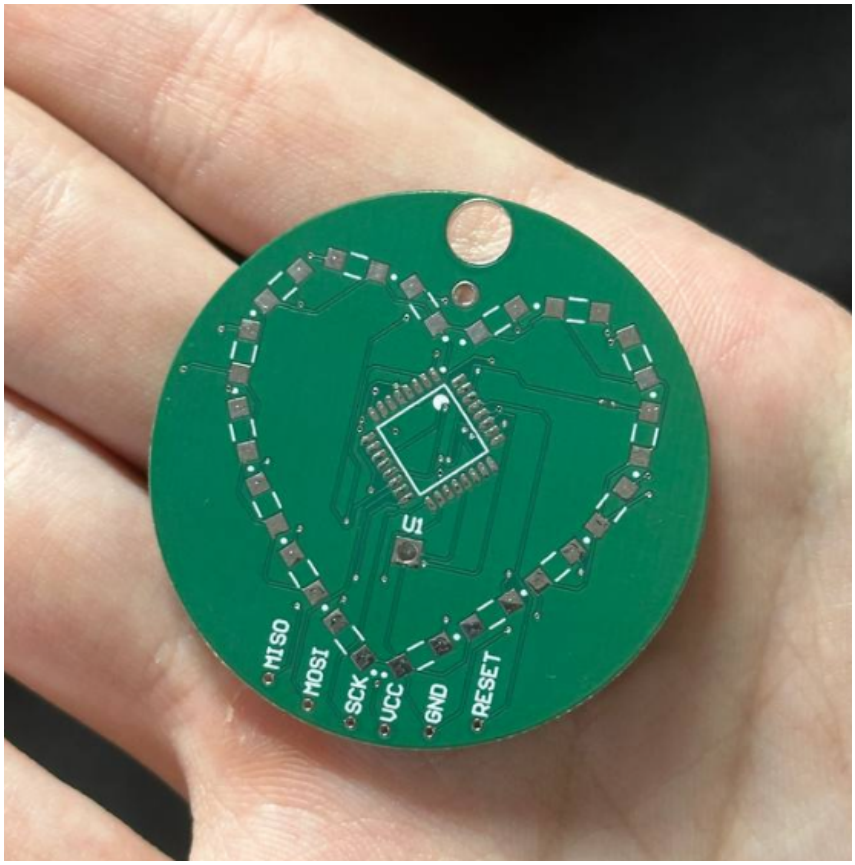
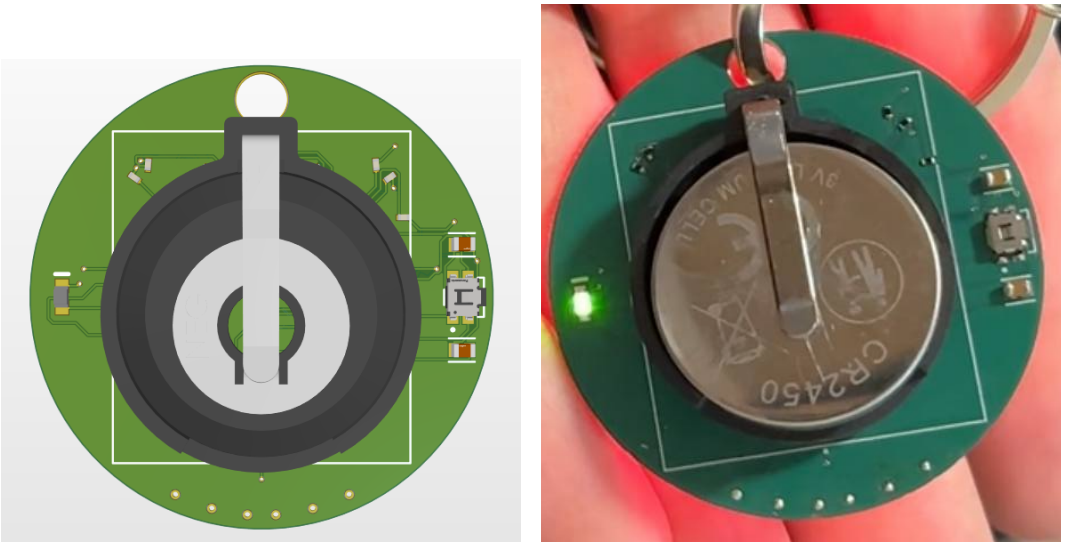


Figure 4: Real Life PCB without components

4.5 Power Supply and Circuit Protection

The heart-shaped PCB is powered by a **3.3V coin cell battery** with a **660mAh** capacity. To ensure smooth power delivery, **ceramic capacitors** were placed near the power rails to filter any noise or fluctuations in voltage.

A **green LED** is used to indicate power status, helping users identify whether the board is active. The LED is connected to the power supply with an appropriate **current-limiting resistor** to prevent excessive power consumption.



(a) PCB Design Back View (b) Real Life Back View

Figure 5: PCB Design Back View andReal Life Back View of the heart-shaped PCB.

4.6 User Interaction

The PCB features a **mode selection button**, which allows the user to toggle between different LED lighting modes. The button is positioned on the back side of the PCB, making it accessible while keeping the front visually clean.

Different lighting modes include:

- Steady glow mode
- Pulsating effect (breathing LED effect)
- Blinking pattern
- Dynamic animations

These modes provide an interactive experience, enhancing the overall appeal of the heart-shaped PCB.

4.7 Challenges Faced During Development

During the design and assembly of the PCB, several challenges arose that required careful problem-solving and iteration to ensure a fully functional and optimized design.

4.7.1 Component Selection and Availability

As this was the first time designing such a project, selecting appropriate components proved to be a significant challenge. Several iterations of the PCB layout were required due to component unavailability or incompatibility. Many components initially chosen were either out of stock or had long lead times, forcing replacements that often required a redesign of the circuit layout. This process consumed considerable time as each change affected routing, power distribution, and mechanical constraints.

4.7.2 PB7 Port Configuration Issue

Another major challenge was related to the microcontrollers **PB7** port. By default, in Arduino-based microcontrollers, PB7 is designated as an **XTAL oscillator port** for an external crystal. However, since the ATmega328P microcontroller used in this project has an internal oscillator, an external crystal was not necessary.

Initially, when uploading the firmware, the LED connected to PB7 did not function as expected. The issue was traced back to the fact that PB7 was not configured as a standard digital output by default. Unlike other GPIO pins, it required an explicit reconfiguration in the firmware.

To resolve this, the microcontrollers configuration had to be modified at a lower level. The fuse bits were checked to ensure that the internal oscillator was being used instead of an external crystal. Additionally, the firmware was modified to explicitly define PB7 as a **digital output**. This required editing the microcontrollers initialization files and modifying the register settings, which was a complex and non-trivial process.

Finding and fixing this issue required extensive debugging, including probing the pin with an oscilloscope to verify its behavior and consulting documentation to understand its default state. Once the modifications were correctly implemented, the LED connected to PB7 began functioning as intended, and the issue was fully resolved.

4.8 Different LEDs Modes

The heart-shaped PCB features different LED animation modes that provide various lighting effects, controlled by the ATmega328P microcontroller. The implementation involves defining LED pins, handling button presses for switching modes, and dynamically changing the LED states in the loop.

4.8.1 LED Pin Definitions

At the beginning of the code, each LED pin is assigned a specific microcontroller pin:

```
#define PINLED0 8
#define PINLED1 5
#define PINLED2 4
#define PINLED3 19
#define PINLED4 18
#define PINLED5 17
#define PINLED6 16
#define PINLED7 15
#define PINLED8 14
#define PINLED9 9
#define PINLED10 21
#define PINLED11 20
#define PINLED12 3
#define PINLED13 1
#define PINLED14 0
#define PINLED15 10
#define PINBUTTON 2
```

These definitions allow easy reference to each LED, simplifying the logic for turning LEDs on and off.

4.8.2 Button Handling

The button is used to switch between LED animation modes. The following variables manage button states and debounce logic:

```
#define BUTTONDEBOUNCE 2
int buttonPressedFor = 0;
int previousState = 0;
int buttonActuallyPressedFor = 0;
int ledState = 0;
```

Here, buttonPressedFor tracks how long the button is held, previousState stores the last button state, and ledState controls the current LED mode.

4.8.3 Turning LEDs On and Off

The turnON and turnOFF functions iterate through all LED pins and set them to HIGH or LOW:

```
void turnON() {
for (int i = 0; i <= 15; i++) {
digitalWrite(PINLED0 + i, HIGH);
}
}
void turnOFF() {
for (int i = 0; i <= 15; i++) {
digitalWrite(PINLED0 + i, LOW);
}
}
```

This approach eliminates redundancy and ensures all LEDs can be controlled efficiently.

4.8.4 Setup Function

The setup() function initializes all LED pins as outputs, ensures they are off at startup, and configures the button pin:

```
void setup() {
for (int i = 0; i <= 15; i++) {
pinMode(PINLED0 + i, OUTPUT);
}
turnOFF();
pinMode(PINBUTTON, INPUT);
}
```

This ensures the board starts with LEDs off and is ready for user interaction.

4.8.5 Loop Function and Button Logic

The loop() function continuously checks for button presses and changes the LED mode accordingly:

```
void loop() {
if(digitalRead(PINBUTTON)) {
buttonPressedFor++;
buttonPressedFor = (buttonPressedFor > BUTTONDEBOUNCE) ? BUTTONDEBOUNCE : buttonPressedFor;
} else {
buttonPressedFor = 0;
}

if(buttonPressedFor >= BUTTONDEBOUNCE) {
if (previousState == 0) {
previousState = 1;
}
} else {
if (previousState == 1) {
previousState = 0;
ledState = (ledState + 1) % 3;
}
}
```

This logic ensures a single press toggles between different LED states.

4.8.6 LED Modes

Depending on the ledState value, different LED animations occur:

```
if (ledState == 0) {
turnOFF();
} else if (ledState == 1) {
turnON();
} else if (ledState == 2) {
heartCouter++;
if (heartCouter > 10) {
heartCouter = 0;
heartPhase = (heartPhase + 1) % 12;
}
}
```

State 0 turns off LEDs, 1 turns all LEDs on, and 2 cycles through different LED phases.

4.8.7 Heart Animation

The heartPhase variable determines which LEDs turn on in each step of the animation:

```
if(heartPhase == 0) {
        digitalWrite(PINLED9, HIGH);
        digitalWrite(PINLED0, HIGH);
      }
      else if (heartPhase == 1){
        digitalWrite(PINLED10, HIGH);
        digitalWrite(PINLED15, HIGH);
      }
      else if (heartPhase == 2){
        digitalWrite(PINLED8, HIGH);
        digitalWrite(PINLED14, HIGH);
      }
      else if (heartPhase == 3){
        digitalWrite(PINLED7, HIGH);
        digitalWrite(PINLED13, HIGH);
      }
      else if (heartPhase == 4){
        digitalWrite(PINLED6, HIGH);
        digitalWrite(PINLED3, HIGH);
        digitalWrite(PINLED9, LOW);
        digitalWrite(PINLED0, LOW);
      }
      else if (heartPhase == 5){
        digitalWrite(PINLED5, HIGH);
        digitalWrite(PINLED12, HIGH);
        digitalWrite(PINLED10, LOW);
        digitalWrite(PINLED15, LOW);
      }
      else if (heartPhase == 6){
        digitalWrite(PINLED4, HIGH);
        digitalWrite(PINLED11, HIGH);
        digitalWrite(PINLED8, LOW);
        digitalWrite(PINLED14, LOW);
      }
      else if (heartPhase == 7){
        digitalWrite(PINLED2, HIGH);
        digitalWrite(PINLED1, HIGH);
        digitalWrite(PINLED6, LOW);
        digitalWrite(PINLED3, LOW);
      }
      else if (heartPhase == 8){
        digitalWrite(PINLED7, LOW);
        digitalWrite(PINLED13, LOW);
        digitalWrite(PINLED0, HIGH);
        digitalWrite(PINLED9, HIGH);
      }
      else if (heartPhase == 9){
        digitalWrite(PINLED5, LOW);
```

```
49          digitalWrite(PINLED12, LOW);
50          digitalWrite(PINLED10, HIGH);
51          digitalWrite(PINLED15, HIGH);
52        }
53        else if (heartPhase == 10){
54          digitalWrite(PINLED4, LOW);
55          digitalWrite(PINLED11, LOW);
56          digitalWrite(PINLED8, HIGH);
57          digitalWrite(PINLED14, HIGH);
58        }
59        else if (heartPhase == 11){
60          digitalWrite(PINLED1, LOW);
61          digitalWrite(PINLED2, LOW);
62          digitalWrite(PINLED7, HIGH);
63          digitalWrite(PINLED13, HIGH);
64        }
65      }
66    }
```

Each step progressively lights up different parts of the heart-shaped LED pattern.

5 Conclusion

The heart-shaped PCB project successfully combined microcontroller programming, LED animation, and PCB design into a compact, aesthetically appealing, and functional electronic system. The primary goal of creating a personalized, interactive LED display was met through careful hardware selection, custom PCB layout, and efficient power management.

Throughout the development process, several challenges were encountered and overcome. One of the major difficulties was component selection, as some parts initially chosen were unavailable or unsuitable for the final design, requiring multiple iterations of the PCB layout. This experience provided valuable insight into supply chain management and the importance of selecting readily available components.

Another critical issue was the configuration of the **PB7** port, which was initially designated as an oscillator pin. Due to the microcontrollers internal oscillator, an external crystal was unnecessary. However, this led to unexpected behavior, requiring manual reconfiguration of the port settings and modification of the firmware to enable PB7 as a digital output. This debugging process highlighted the importance of understanding low-level microcontroller configurations and register manipulations.

The project also involved complex power management considerations. By implementing appropriate current-limiting resistors and capacitors for noise filtering, the circuit was optimized for stable and efficient operation. The power supply, a 3.3V coin cell battery, was carefully chosen to balance portability with sufficient runtime. Detailed power consumption calculations ensured that the system could operate for an extended period while maintaining consistent LED performance.

The software development process included implementing multiple LED animation modes, controlled via a tactile button. A robust button debounce algorithm was employed to prevent erroneous state changes, ensuring smooth user interaction. The heart animation sequence was carefully designed to create a visually engaging effect, making the PCB both functional and decorative.

The final assembly process was done manually, requiring precise soldering techniques due to the small size of the surface-mount components. The use of professional tools, such as a microscope, ensured a high-quality build and reliable electrical connections. Despite the challenges of assembling 0402-sized resistors and tightly packed components, the final product was successfully completed with all intended features working as expected.

In conclusion, this project served as an excellent learning experience in embedded systems, PCB design, and real-world hardware development. The knowledge gained in component selection, circuit optimization, firmware debugging, and assembly techniques will be invaluable for future projects. The final heart-shaped PCB is not only a technical achievement but also a meaningful, interactive electronic device that demonstrates the integration of hardware and software in a creative and practical way.

Appendix

The full Arduino code is listed below:

```
#define PINLED0 8
#define PINLED1 5
#define PINLED2 4
#define PINLED3 19
#define PINLED4 18
#define PINLED5 17
#define PINLED6 16
#define PINLED7 15
#define PINLED8 14
#define PINLED9 9
#define PINLED10 21
#define PINLED11 20
#define PINLED12 3
#define PINLED13 1
#define PINLED14 0
#define PINLED15 10

#define PINBUTTON 2

#define BUTTONDEBOUNCE 2

int buttonPressedFor = 0;
int previousState = 0;
int buttonActuallyPressedFor = 0;
int ledState = 0;

int heartCouter = 0;
int heartPhase = 0;

void turnON() {
  digitalWrite(PINLED0, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED1, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED2, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED3, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED4, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED5, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED6, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED7, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED8, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED9, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED10, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED11, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED12, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED13, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED14, HIGH);  // turn the LED on (HIGH is the voltage level)
  digitalWrite(PINLED15, HIGH);  // turn the LED on (HIGH is the voltage level)
}

```

```
49  void turnOFF() {
50    digitalWrite(PINLED0, LOW);  // turn the LED on (HIGH is the voltage level)
51    digitalWrite(PINLED1, LOW);  // turn the LED on (HIGH is the voltage level)
52    digitalWrite(PINLED2, LOW);  // turn the LED on (HIGH is the voltage level)
53    digitalWrite(PINLED3, LOW);  // turn the LED on (HIGH is the voltage level)
54    digitalWrite(PINLED4, LOW);  // turn the LED on (HIGH is the voltage level)
55    digitalWrite(PINLED5, LOW);  // turn the LED on (HIGH is the voltage level)
56    digitalWrite(PINLED6, LOW);  // turn the LED on (HIGH is the voltage level)
57    digitalWrite(PINLED7, LOW);  // turn the LED on (HIGH is the voltage level)
58    digitalWrite(PINLED8, LOW);  // turn the LED on (HIGH is the voltage level)
59    digitalWrite(PINLED9, LOW);  // turn the LED on (HIGH is the voltage level)
60    digitalWrite(PINLED10, LOW);  // turn the LED on (HIGH is the voltage level)
61    digitalWrite(PINLED11, LOW);  // turn the LED on (HIGH is the voltage level)
62    digitalWrite(PINLED12, LOW);  // turn the LED on (HIGH is the voltage level)
63    digitalWrite(PINLED13, LOW);  // turn the LED on (HIGH is the voltage level)
64    digitalWrite(PINLED14, LOW);  // turn the LED on (HIGH is the voltage level)
65    digitalWrite(PINLED15, LOW);  // turn the LED on (HIGH is the voltage level)
66  }
67
68  void HeartMovement() {
69
70  }
71
72  void setup() {
73    // initialize digital pin LED_BUILTIN as an output.
74    pinMode(PINLED0, OUTPUT);
75    pinMode(PINLED1, OUTPUT);
76    pinMode(PINLED2, OUTPUT);
77    pinMode(PINLED3, OUTPUT);
78    pinMode(PINLED4, OUTPUT);
79    pinMode(PINLED5, OUTPUT);
80    pinMode(PINLED6, OUTPUT);
81    pinMode(PINLED7, OUTPUT);
82    pinMode(PINLED8, OUTPUT);
83    pinMode(PINLED9, OUTPUT);
84    pinMode(PINLED10, OUTPUT);
85    pinMode(PINLED11, OUTPUT);
86    pinMode(PINLED12, OUTPUT);
87    pinMode(PINLED13, OUTPUT);
88    pinMode(PINLED14, OUTPUT);
89    pinMode(PINLED15, OUTPUT);
90    turnOFF();
91    //digitalWrite(PINLED9, HIGH);
92    //digitalWrite(PINLED0, HIGH);
93
94    pinMode(PINBUTTON, INPUT);
95
96    buttonPressedFor = 0;
97    previousState = 0;
98    ledState = 0;
99
100   heartCouter = 0;
```

```
}

void loop() {
  // 11
  if(digitalRead(PINBUTTON)) {
    buttonPressedFor++;
    buttonPressedFor = (buttonPressedFor > BUTTONDEBOUNCE) ? BUTTONDEBOUNCE : buttonPressedFor;
    //turnON();
  }
  else {
    buttonPressedFor = 0;
    //turnOFF();
  }

  if(previousState == 1) {
    buttonActuallyPressedFor++;
    buttonActuallyPressedFor = (buttonActuallyPressedFor > 10000) ? 10000 : buttonActuallyPressedFor;
  }

  if(buttonPressedFor >= BUTTONDEBOUNCE) {
    // button is actually pressed
    if (previousState == 0) {
      // unpressed -> pressed
      previousState = 1;

      /*if (ledState == 0) {
        //turnON();
        ledState = 1;
      }
      else {
        //turnOFF();
        ledState = 0;
      }*/

    }
  }
  else {
    if (previousState == 1) {
      // pressed -> unpressed
      previousState = 0;
      if(buttonActuallyPressedFor > 100) {
        // long press
        // ...
        if (ledState == 0) {
          ledState = 0;
        }
        else if (ledState == 1) {
          ledState = 2;

          heartCouter = 0;
          turnOFF();
          digitalWrite(PINLED9, HIGH);
```

```
          digitalWrite(PINLED0, HIGH);
        }
        else if (ledState == 2) {
          ledState = 1;
        }
      }
      else {
        // short press
        if (ledState == 0) {
          ledState = 1;
        }
        else if (ledState == 1) {
          ledState = 0;
        }
        else if (ledState == 2) {
          ledState = 0;
        }
      }
      buttonActuallyPressedFor = 0;
    }
  }

  if (ledState == 0) {
    turnOFF();
  }
  else if (ledState == 1){
    turnON();
  }
  else if (ledState == 2) {
    heartCouter++;
    if (heartCouter > 10) {
      heartCouter = 0;


      heartPhase++;
      heartPhase = (heartPhase > 11) ? 4 : heartPhase;

if(heartPhase == 0) {
        digitalWrite(PINLED9, HIGH);
        digitalWrite(PINLED0, HIGH);
      }
      else if (heartPhase == 1){
        digitalWrite(PINLED10, HIGH);
        digitalWrite(PINLED15, HIGH);
      }
      else if (heartPhase == 2){
        digitalWrite(PINLED8, HIGH);
        digitalWrite(PINLED14, HIGH);
      }
      else if (heartPhase == 3){
        digitalWrite(PINLED7, HIGH);
        digitalWrite(PINLED13, HIGH);
```

```
205       }
206       else if (heartPhase == 4){
207         digitalWrite(PINLED6, HIGH);
208         digitalWrite(PINLED3, HIGH);
209         digitalWrite(PINLED9, LOW);
210         digitalWrite(PINLED0, LOW);
211       }
212       else if (heartPhase == 5){
213         digitalWrite(PINLED5, HIGH);
214         digitalWrite(PINLED12, HIGH);
215         digitalWrite(PINLED10, LOW);
216         digitalWrite(PINLED15, LOW);
217       }
218       else if (heartPhase == 6){
219         digitalWrite(PINLED4, HIGH);
220         digitalWrite(PINLED11, HIGH);
221         digitalWrite(PINLED8, LOW);
222         digitalWrite(PINLED14, LOW);
223       }
224       else if (heartPhase == 7){
225         digitalWrite(PINLED2, HIGH);
226         digitalWrite(PINLED1, HIGH);
227         digitalWrite(PINLED6, LOW);
228         digitalWrite(PINLED3, LOW);
229       }
230       else if (heartPhase == 8){
231         digitalWrite(PINLED7, LOW);
232         digitalWrite(PINLED13, LOW);
233         digitalWrite(PINLED0, HIGH);
234         digitalWrite(PINLED9, HIGH);
235       }
236       else if (heartPhase == 9){
237         digitalWrite(PINLED5, LOW);
238         digitalWrite(PINLED12, LOW);
239         digitalWrite(PINLED10, HIGH);
240         digitalWrite(PINLED15, HIGH);
241       }
242       else if (heartPhase == 10){
243         digitalWrite(PINLED4, LOW);
244         digitalWrite(PINLED11, LOW);
245         digitalWrite(PINLED8, HIGH);
246         digitalWrite(PINLED14, HIGH);
247       }
248       else if (heartPhase == 11){
249         digitalWrite(PINLED1, LOW);
250         digitalWrite(PINLED2, LOW);
251         digitalWrite(PINLED7, HIGH);
252         digitalWrite(PINLED13, HIGH);
253       }
254     }
255   }
256
```

```
257   delay(1);                       // wait for a second
258   // digitalWrite(PINLED, LOW);   // turn the LED off by making the voltage LOW
259   // delay(1000);                      // wait for a second
260 }
```