# Super Mario

Custom Project Final Report

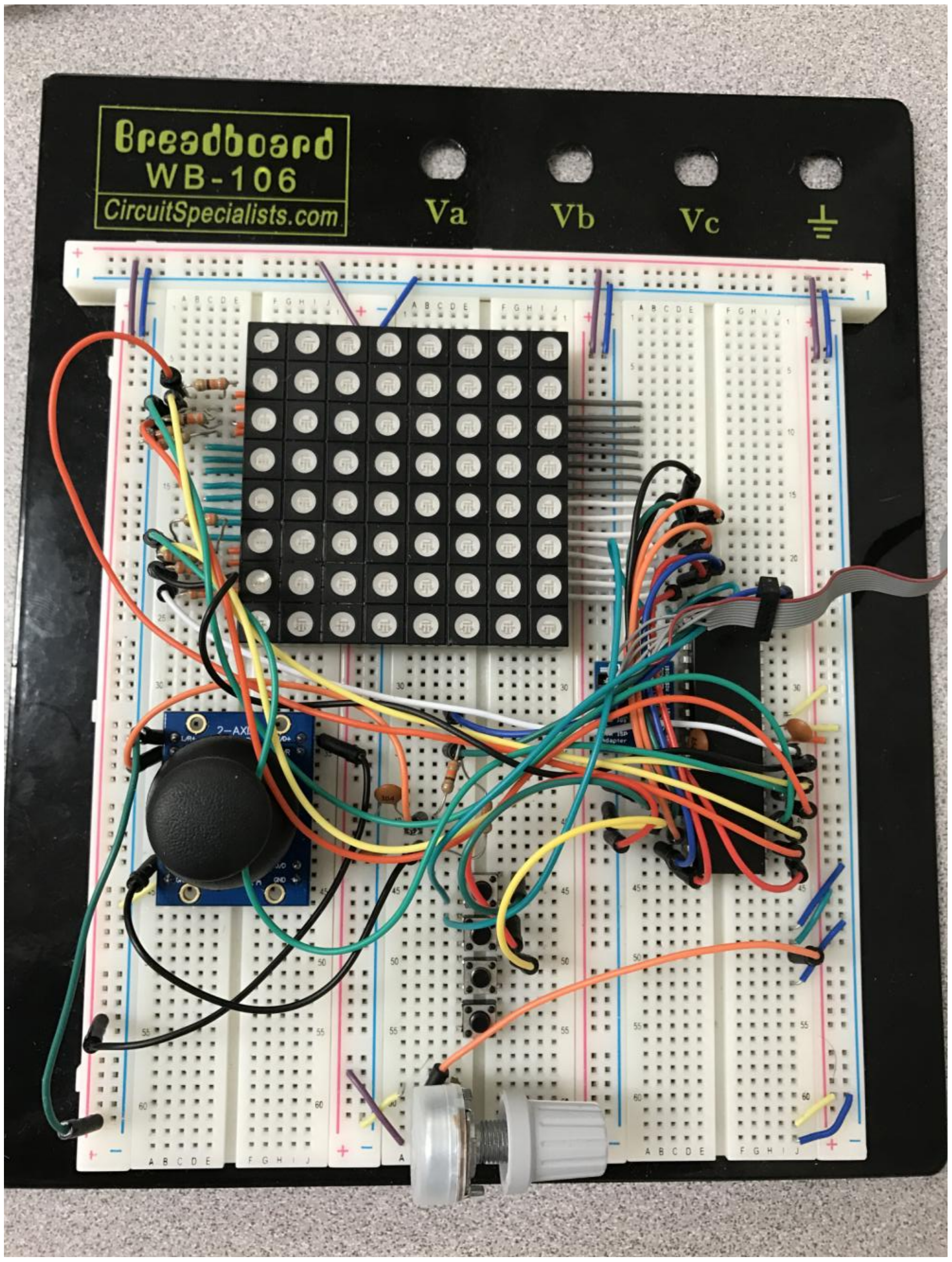
Spring 2017

Yash Kelkar

# Table of Contents

## Introduction

Super Mario is a two-dimensional platform game. The play controls Mario with a joystick controlling him by either moving left, moving right, diagonally right, diagonally left, or jumping straight up. The goal is the reach the castle at the end of the level which indicates that you have won and the game is over.



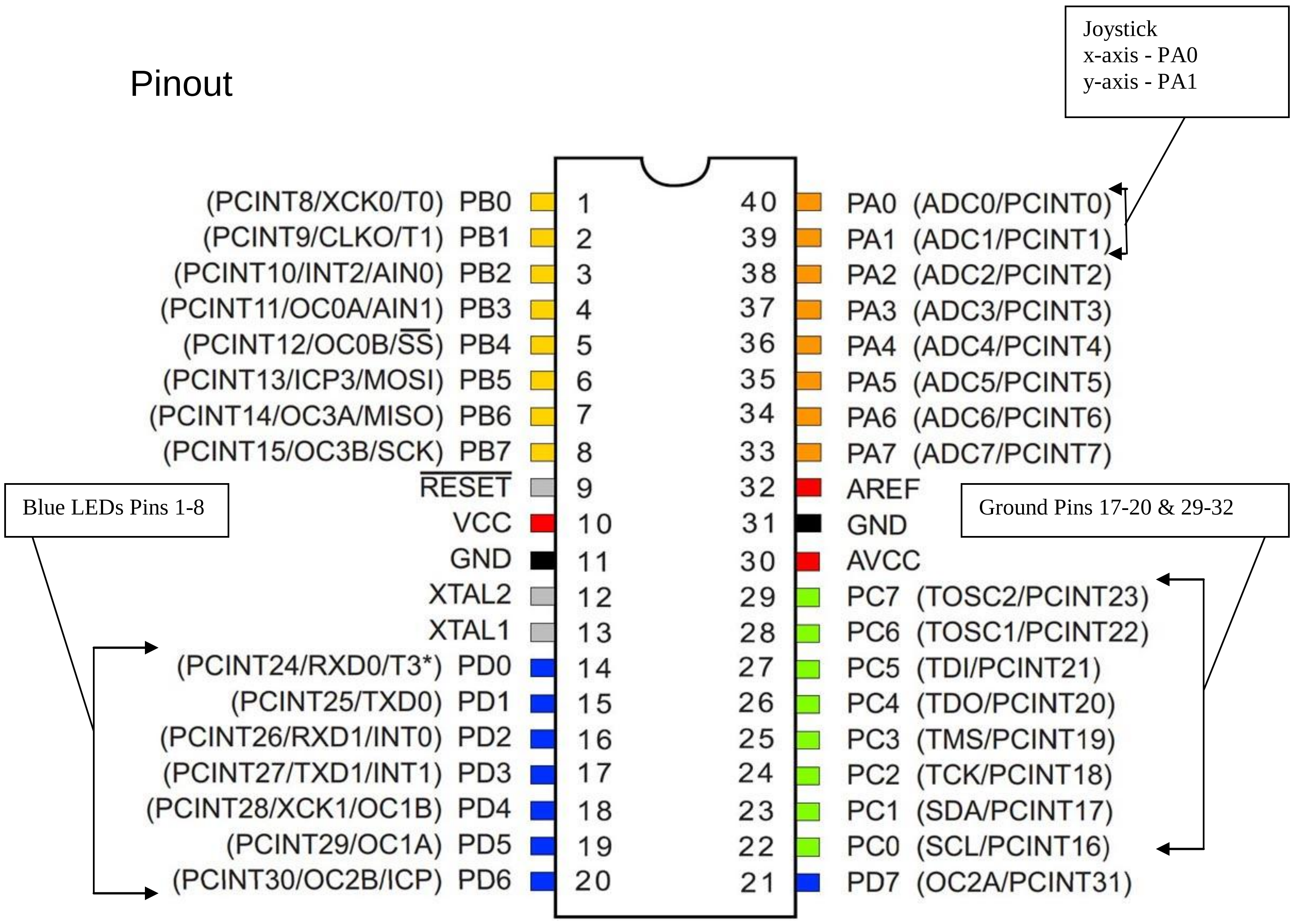
# Hardware

## Parts List

The hardware that was used in this design is listed below. The equipment that was not taught in this course has been bolded. Include part numbers when available.

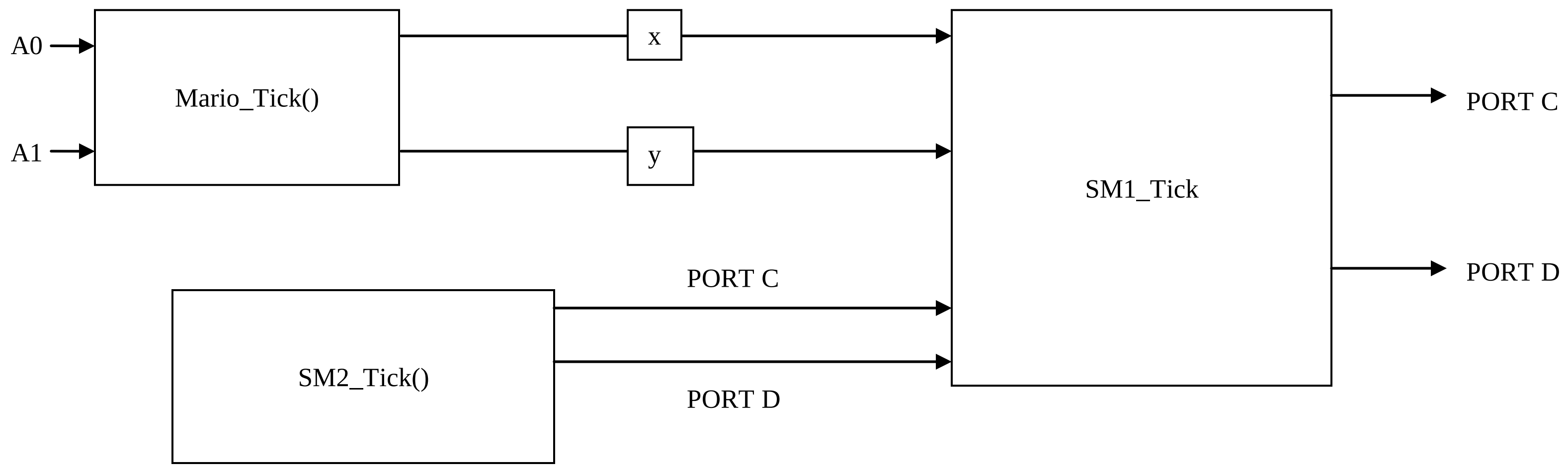
- ATMega1284p microcontroller
- **8x8 LED matrix**
- **Parallax 2-Axis Joystick**

## Pinout


Joystick
x-axis - PA0
y-axis - PA1
Blue LEDs Pins 1-8
Ground Pins 17-20 & 29-32
(PCINT8/XCK0/T0) PB0 1
(PCINT9/CLKO/T1) PB1 2
(PCINT10/INT2/AIN0) PB2 3
(PCINT11/OC0A/AIN1) PB3 4
(PCINT12/OC0B/SS) PB4 5
(PCINT13/ICP3/MOSI) PB5 6
(PCINT14/OC3A/MISO) PB6 7
(PCINT15/OC3B/SCK) PB7 8
RESET 9
VCC 10
GND 11
XTAL2 12
XTAL1 13
(PCINT24/RXD0/T3*) PD0 14
(PCINT25/TXD0) PD1 15
(PCINT26/RXD1/INT0) PD2 16
(PCINT27/TXD1/INT1) PD3 17
(PCINT28/XCK1/OC1B) PD4 18
(PCINT29/OC1A) PD5 19
(PCINT30/OC2B/ICP) PD6 20
40 PA0 (ADC0/PCINT0)
39 PA1 (ADC1/PCINT1)
38 PA2 (ADC2/PCINT2)
37 PA3 (ADC3/PCINT3)
36 PA4 (ADC4/PCINT4)
35 PA5 (ADC5/PCINT5)
34 PA6 (ADC6/PCINT6)
33 PA7 (ADC7/PCINT7)
32 AREF
31 GND
30 AVCC
29 PC7 (TOSC2/PCINT23)
28 PC6 (TOSC1/PCINT22)
27 PC5 (TDI/PCINT21)
26 PC4 (TDO/PCINT20)
25 PC3 (TMS/PCINT19)
24 PC2 (TCK/PCINT18)
23 PC1 (SDA/PCINT17)
22 PC0 (SCL/PCINT16)
21 PD7 (OC2A/PCINT31)

## Software

The software designed for this project was implemented using the PES standard. The overall design as a task diagram is included below.



Now I will write a short description of the tasks in the project. The appendix will include SM's that I designed.

## Complexities

### Completed Complexities:

- Integrating and calibrating the joystick
- Game logic of Super Mario
  - Mario can move up, down, left, right, diagonally right, diagonally left
  - If Mario is caught between the game platform when it shifts off the LED matrix then he dies
  - The game resets when he dies
  - The game is over when the castle is reached.
- Creating custom game platform on the LED matrix
- Interfacing one LED (Mario) with game platform so that Mario is able the jump on obstacles and moves with them. Gives the physical aspect of the game.

Incomplete complexities:

- Mario Theme Song on Speaker: I did not complete this because I was not able to get the Mario Theme song to work. I looked it up online, but failed to integrate it into my project.
- Score Keeper on LCD: I did not complete this because the rest of the project took longer than I thought and did not get to it.

## Youtube Link

- ➢ https://www.youtube.com/watch?v=UI3YanLzr5o

## Known Bugs and Shortcomings

- The game starts right away when you turn it on.

- Towards the end of the level the lights start flashing. I think the cause is the time overlaps within the code. I will start debugging within my SM1_Tick() function because this is where I store and display the values for the game on the LED matrix.

## Future work

In this section pretend for a moment that you were going to continue working on the project. What would be the next feature you would add?

The next features I would add would be the Mario Theme Song, the score on an LCD display, more levels, and make use of the possible colors the LED can display. For example, Mario would be red and then different parts of the game would be different colors such as green for the ground.

## References

https://ilearn.ucr.edu/bbcswebdav/pid-2932365-dt-content-rid-16110947_1/courses/CS_120B_001_17S/eecs120b_labX_LED_matrix-2.pdf

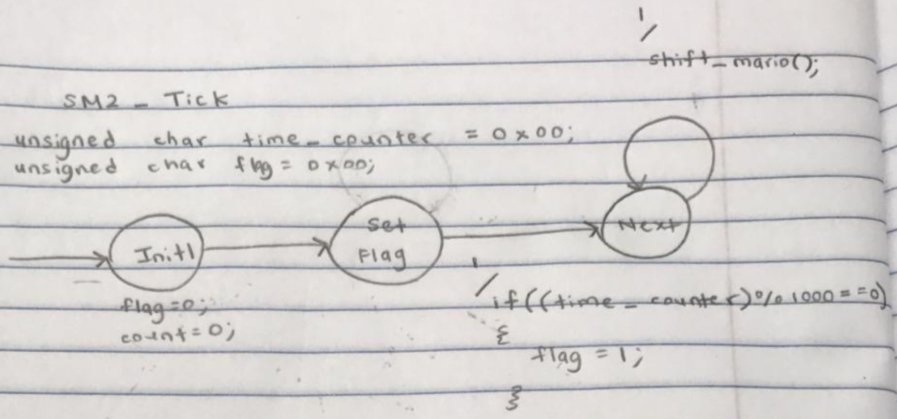
http://extremeelectronics.co.in/avr-tutorials/interfacing-analog-joystick-with-avr-atmega32/

## Appendix

Include images of all of the SM's that you have designed and any other work that you think is relevant to this project.

SM2 _ Tick

```
unsigned char time_counter = 0x00;
unsigned char flag = 0x00;
```



```
void shift_mario()
{
        if ((time_counter) % 1000 == 0)
        {
                row[1] = row[1] >>1;
                row[2] = row[2] >>1;
                ...
                row[6] = row[6] >>1;

                if ( count >2)
                {
                        row[7] = row[7] >>1;
                }
                ....
                if (count >38)
                {
                        row[32] = row[32] >>1;
                }
        }
}
```

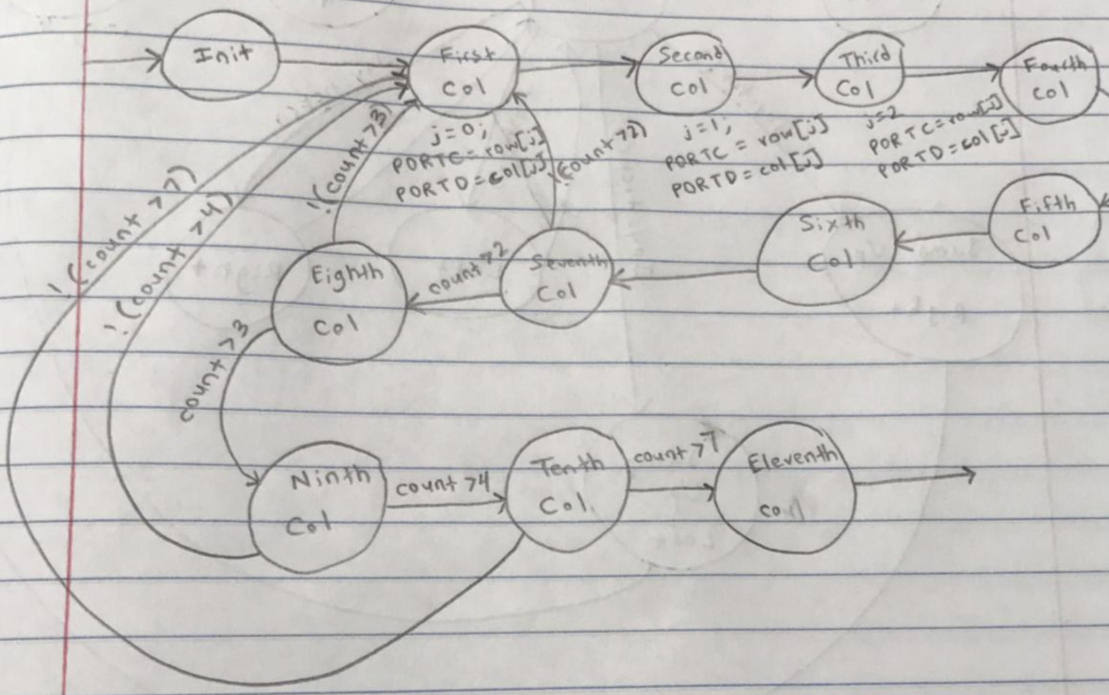
```
static unsigned char row[];
static unsigned char col[];

unsigned char j = 0x00
```

SM1 _ Tick

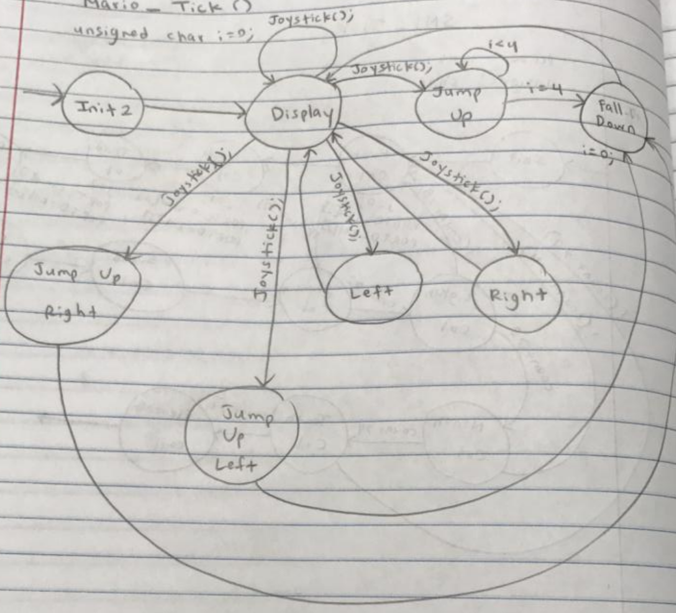
Period = 1 ms

```
unsigned char i;
```



• This pattern continues until column 32 ( Thirty - Second _ Col)

```
uint16_t x, y;
x = readadc(0);
y = readadc(1);
Mario_Tick()
unsigned char i=0;
```



```
void Joystick()
{
    if (x > 600)
    {
        if (x > 575)
        {
            Jump Up Right
        }
        else if ( x < 500)
        {
            Jump Up Left
        }
        else
        {
            Jump Up
        }
}
```