Kunal Mukherjee

ee 354

Project 2: hour-glass, WATER-FALL & PACHINKo simulation EE 354 12/1/2018



Contents

[Novel Feature 2](#_Toc531444158)

[Working Feature 3](#_Toc531444159)

[Safety, reliability, economic, manufacturability and environmental factors 4](#_Toc531444160)

[Power Measurement 5](#_Toc531444161)

[Hardware Documentation 5](#_Toc531444162)

[Software Documentation 9](#_Toc531444163)

[**Pseudocode** 9](#_Toc531444164)

[**Actual Code** 13](#_Toc531444165)

[Figure 1: Output Pin Mapping 5](file:///C:\Users\kunmu\Documents\Kunal\UE%20courses\EE-354\project2\documentation.docx#_Toc531440242)

[Figure 2: INPUT pin Mapping 6](file:///C:\Users\kunmu\Documents\Kunal\UE%20courses\EE-354\project2\documentation.docx#_Toc531440243)

[Figure 3: Power Supply 6](#_Toc531440244)

[Figure 4: Length of Box 7](#_Toc531440245)

[Figure 5: Height of Box 7](#_Toc531440246)

[Figure 6: Width of Box 8](#_Toc531440247)

[Figure 7: Internal Picture of Box 8](#_Toc531440248)

# Novel Feature

My hourglass has three novel features:

1. The hour glass does not let the sand fall until the tilt of the box is more that 45°. This is a feature that can be seen in normal hour glass also, that if you don’t tilt the glass more than till a limit the sands would move but not fall. I tried to emulate that effect. This happens on both axes.
2. The hourglass, waterfall and pachinko have a pause setting, that if you move the switch will be paused at that instant. Even if you can move the hourglass, waterfall and pachinko whichever direction you want, when you resume it the sands will flow according to gravity.
3. The hourglass, waterfall and pachinko work continuously, that you can keep rotating the hourglass and the sands will flow, without the need for resetting.
4. Pachinko is made in a way, that after the sands fill up the entire led matrix, it will automatically restart, by clearing the led matrix.

# Working Feature

The physical sheet has been provided. The following features shown working:

1. Does not rattle when shaken
2. Project uses ARM stm32F446 Nucleo board
3. Batteries can be changed without disassembly
4. The project is fully self-contained (uses batteries)
5. There are an on/off and reset switch
6. Software has a mix of C and assembly code
7. Successfully does hourglass, waterfall and pachinko simulation
8. Successfully works when rotated

# Safety, reliability, economic, manufacturability and environmental factors

**Safety:**

The box is made with plastic and every wire ending is insulated with hot glue. The lid is water proof and is also, made with plastic. So, there is no conduction of current possible from the box.

**Reliability:**

The hour glass, only function, the sands flow, if it is above a particular tilt. You can rely on the hourglass to get see the direction of gravity as well as the orientation of the box, that is the sand is flowing then the box is at a tile more than 45° from the x-axis.

**Economic:**

The entire board has constructed cost is less than $100.So, it is extremely cheap to manufacture the hour glass, if the user has access to the hardware and software documentation.

**Manufacturability:**

The entire software, hardware documentation as well as the source code is provided. With this project documentation guide, anyone with basic engineering knowledge can recreate my hour glass.

**Environmental:**

Rechargeable 9V battery can be used for this project, so it saved the environment as we are reusing battery. The project does not emit any harmful gas or chemical substance.

# Power Measurement

The power was calculated with the formulae P = V^2 / R. First, the resistance of the ARM board was measured along with the LED, accelerometer and battery connected. The voltage was already known to be 5V, as that was the voltage being regulated by the 5V regulator.

Power came out to be, P = (5V^2 / (30 Ω) => p = 0.833 W. But the battery was supplying 9V and, we were regulating it to 5V, so it is wasting a lot of energy. The power wasted calculation is: P = ΔV⋅I = (9V−5V) ⋅ 1A = 4W.

# Hardware Documentation

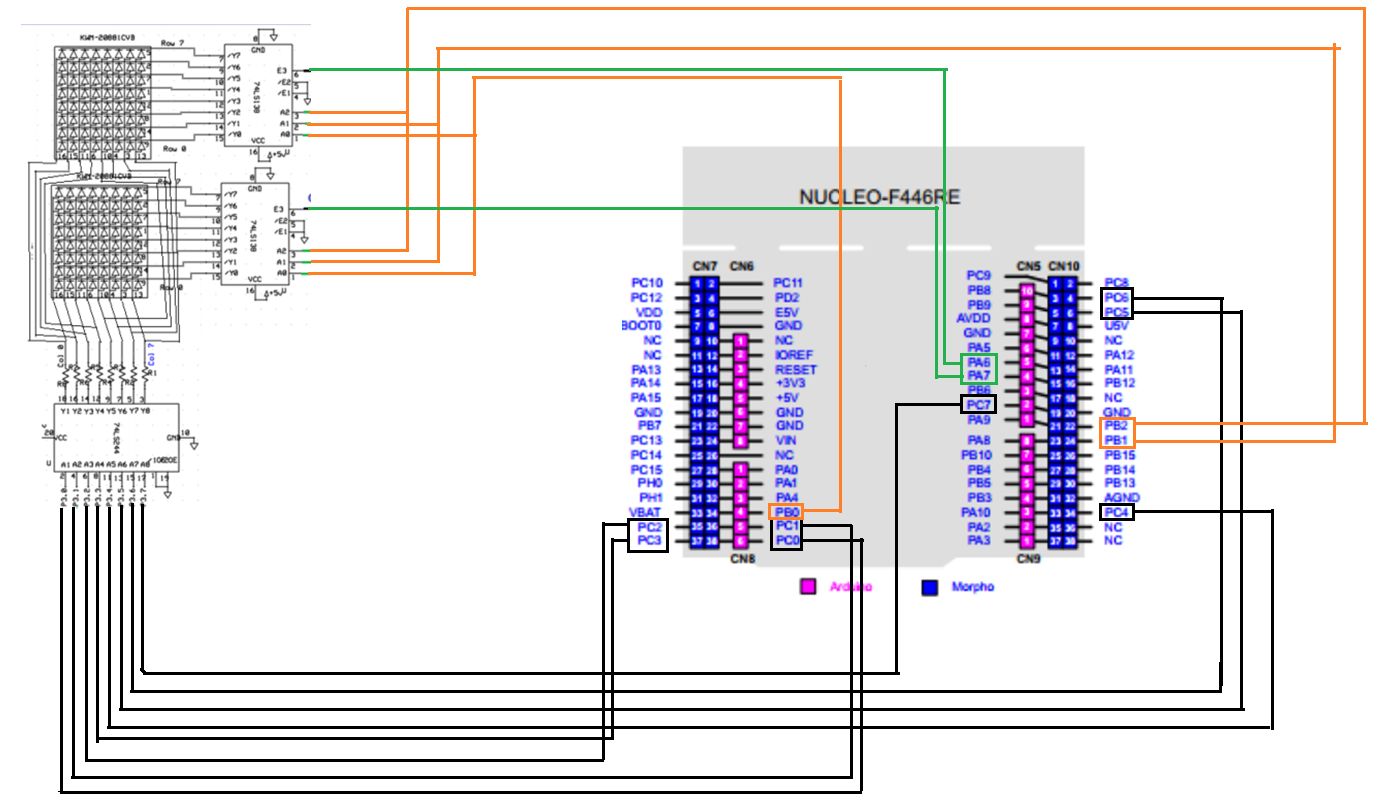


Figure 1: Output Pin Mapping

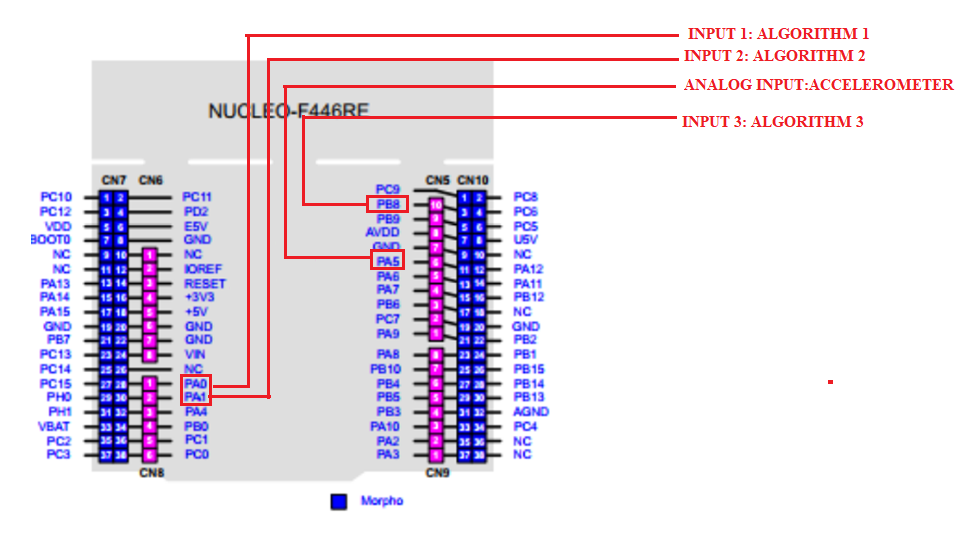


Figure 2: INPUT pin Mapping

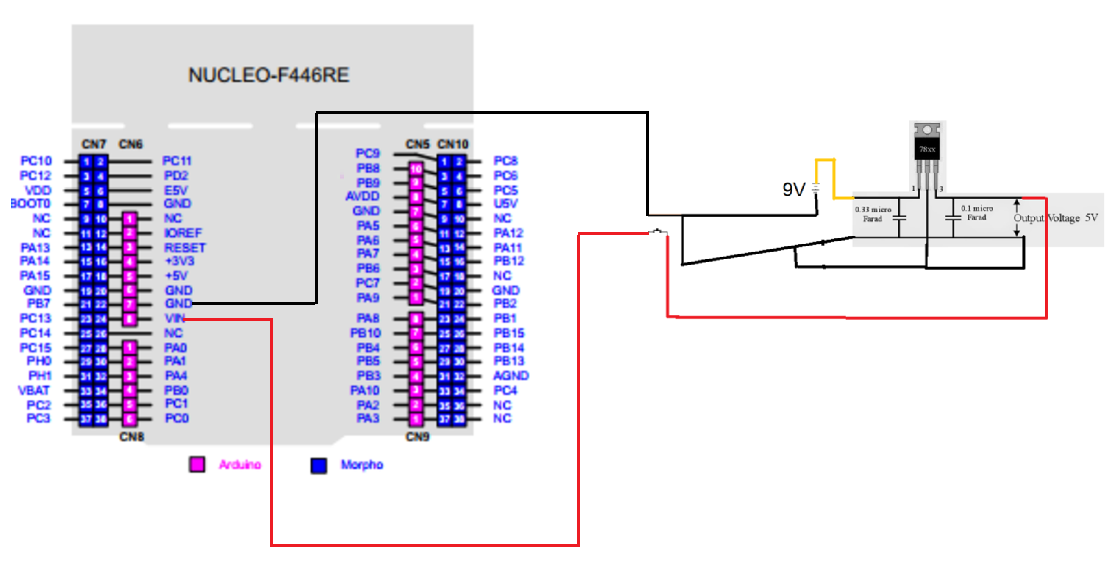


Figure 3: Power Supply



Figure 4: Length of Box



Figure 5: Height of Box



Figure 6: Width of Box

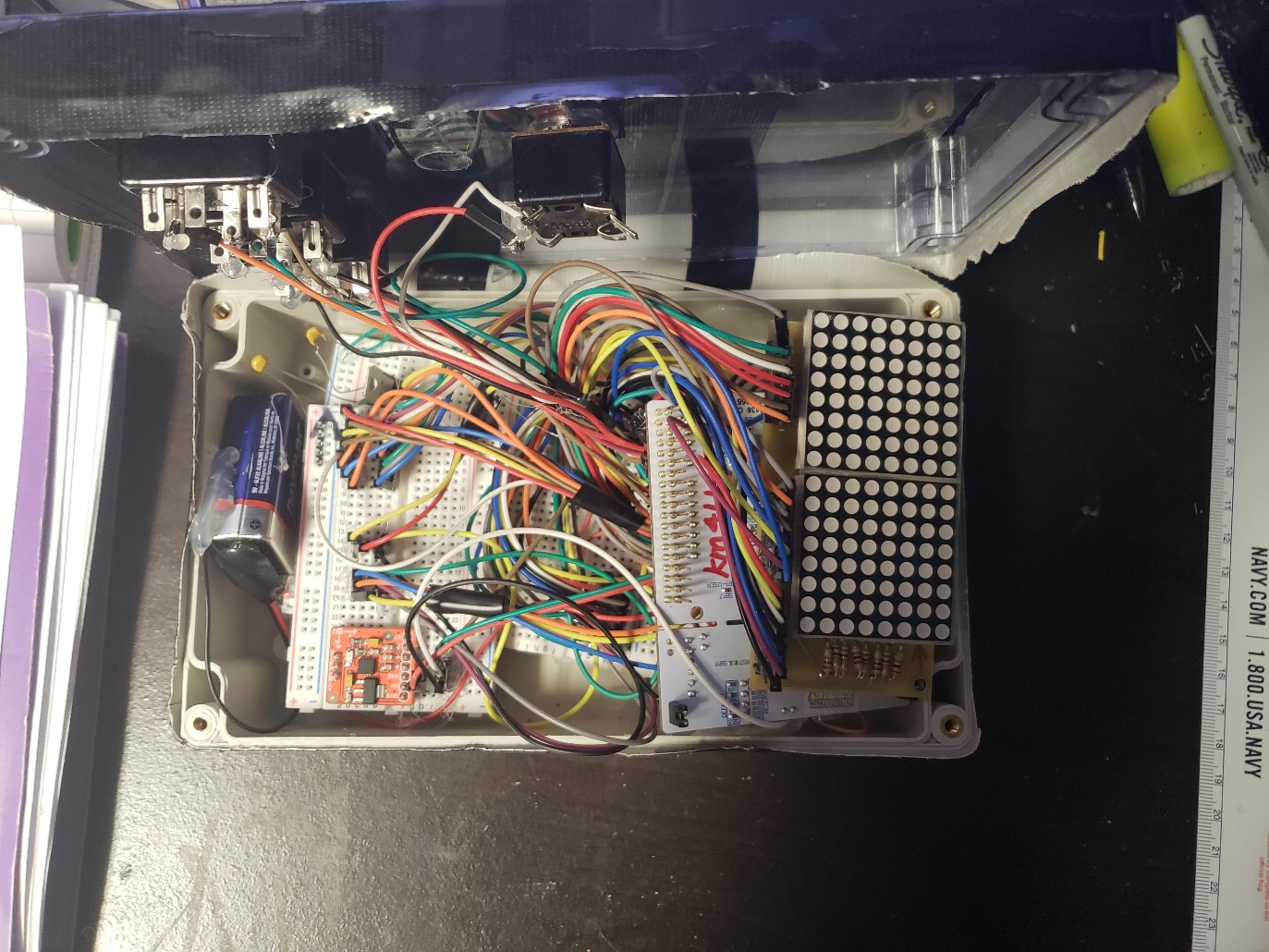


Figure 7: Internal Picture of Box

# Software Documentation

## **Pseudocode**

* Initialize global interrupt variable for Timer 2 multiplexing
* Initialize global variable constants
  + Unsigned char Down
  + Unsigned char Up
  + Unsigned char Flat
* Initialize global variables
  + Unsigned char led[16]
  + Unsigned refresh row
  + Int pachinkoround flag
  + Flag
  + Flag1
  + Flag2
  + Flag3
* Main code
* {
* Enabling Clock bits: gpioa, gpiob, gpioc, peripheral timer 2
* Enable i/o bits
* Setup gpio pins
* Setup gioi pins
* Setup accelerometer
* Setup interrupt bits
* Setup timer 2 bits
* Initialize local variables
  + Clear LED memory
* While(not stopper)
* {
* if(flag)
  + { restart timer
  + Flag = 0}
* if (algorithm 1 select)
* {
  + If(flag1 = 0)
  + {
    - Flag1 = 1
    - Flag2 = 0
    - Flag3 = 0
    - Clear led memory map
    - Place hour glass outline in memory map}
* If(getupDn() == down)
* {
  + For (ledrow =15;ledrow >=0; ledrow++){
  + For(ledcol = 0; ledcol < 8; ledcol++){
  + DOLED1(ledrow,ledcol);}}}
* Else{
  + For (ledrow =0;ledrow <16; ledrow++){
  + For(ledcol = 0; ledcol < 8; ledcol++){
  + DOLED1(ledrow,ledcol);}}}
* }
* if (algorithm 2 select)
* {
  + If(flag1 = 0)
  + {
    - Flag1 = 0
    - Flag2 = 1
    - Flag3 = 0
    - Clear led memory map }
* If(getupDn() == down)
* {
  + Get random column
  + Turn on led on the random column
  + For (ledrow =15;ledrow >=0; ledrow++){
  + For(ledcol = 0; ledcol < 8; ledcol++){
  + DOLED2(ledrow,ledcol);}}}
* Else{
  + Get random column
  + Turn on led on the random column
  + For (ledrow =0;ledrow <16; ledrow++){
  + For(ledcol = 0; ledcol < 8; ledcol++){
  + DOLED2(ledrow,ledcol);}}}
* }
* if (algorithm 3 select)
* {
  + If(flag3 = 0)
  + {
    - Flag1 = 0
    - Flag2 = 0
    - Flag3 = 1
    - Clear led memory map
    - Place water glass outline in memory map}
* If(getupDn() == down)
* {
  + For (ledrow =15;ledrow >=0; ledrow++){
  + For(ledcol = 0; ledcol < 8; ledcol++){
  + DOLED3(ledrow,ledcol);}}}
* Else{
  + For (ledrow =0;ledrow <16; ledrow++){
  + For(ledcol = 0; ledcol < 8; ledcol++){
  + DOLED3(ledrow,ledcol);}}}
* }
* }
* Timer 2 irqhandler {
  + Clear gpioc
  + D = led[refreshrow]
  + Gpioc |= d
  + Convert refreshrow to port bits for decoder
    - Output encoder
  + Update refresh row}
* Output encoder {
  + Selecting the correct decoder
  + Sending the correct selecting to the decoder
  + Update refresh row}
* Void doled1 (r,c){
* If( getupdn == down){
  + Look at neighbours below (r,c)
  + Move led at (r,c) according to sandfalling algorithm
  + }
* Else{
  + Look at neighbours below (r,c)
  + Move led at (r,c) according to sandfalling algorithm
  + }
* }
* Void doled2 (r,c){
* If( getupdn == down){
  + Move led at (r,c) towards gravity without going over led
  + }
* Else{
  + Move led at (r,c) towards gravity without going over led
  + }
* }
* Void doled3 (r,c){
* If( getupdn == down){
  + Look at neighbours below (r,c)
  + Move led at (r,c) according to waterfalling algorithm
  + }
* Else{
  + Look at neighbours below (r,c)
  + Move led at (r,c) according to waterfalling algorithm
  + }
* }

## **Actual Code**

