ដេប៉ាតឺម៉ង: ពត៌មានវិទ្យា
Subject : Computer Architicture
Project : Arduino Game
បង្រៀនដោយលោកគ្រូ : Ouk Polyvann

ក្រុមទី ១

ល.រ	ឈ្មោះ	ពិន្ទុ	តួនាទី	ភាគរយនៃការយល់ដឹង
១	ស៊ូ ចាន់រ៉ូជែម		Code, Slide, Simulator, hardware	
២	សាំង មិញស៊ឺ			
៣	វិន ម៉េងឡុង			
៤	លី តុលា			
៥	វិបុល សុខលីម			
៦	លី ម៉េងហ៊ាង		ជួយភ្ជាប់LCD	
៧	ស៊ន ចាន់ឆៃហុង			
៨	វ៉ាង វាន់ថេន			

INTRODUCTION

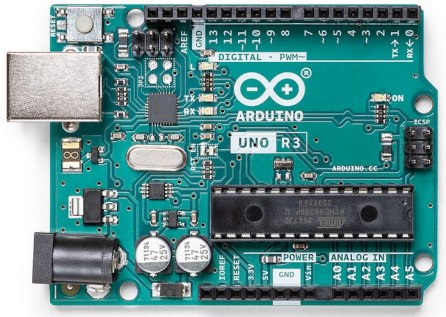
នៅពេលបច្ចុប្បន្ន យើងឃើញថាបច្ចេកវិទ្យារ៉ូបូតកំពុងមានការរីកចម្រើនយ៉ាងខ្លាំង គេបានយកបច្ចេកវិទ្យានេះទៅបំពាក់លើរយន្ត ដើម្បីអោយរថយន្តមានភាពឆ្លាតវៃអាចដំណើរការដោយការបញ្ជារតាម តែឡេ តាមទូរស្ទព័ជាដើម ដើម្បីសម្រួលដល់ការប្រើប្រាស់ នឹងមានភាពងាយស្រួលដល់មនុស្សយើងគ្រប់វិស័យ៖

- វិស័យកសិកម្ម
- វិស័យឧស្សាហកម្ម
- វិស័យអប់រំ
- វិស័យយោធា

Arduino LCD Run Game

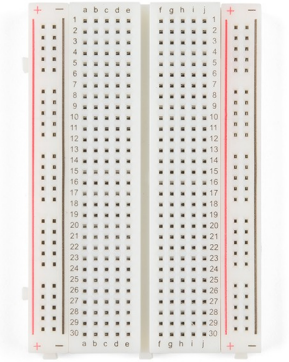
Aruino LCD Game require:

-Arduino Board

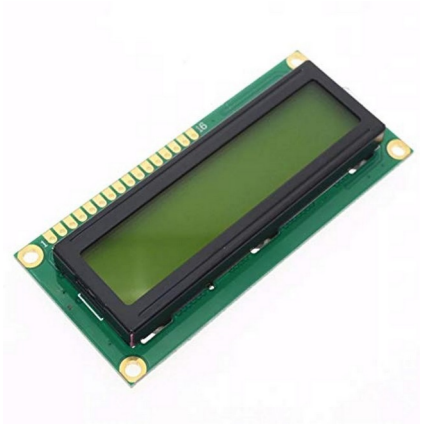


-10K ohm resistor

-breadboard



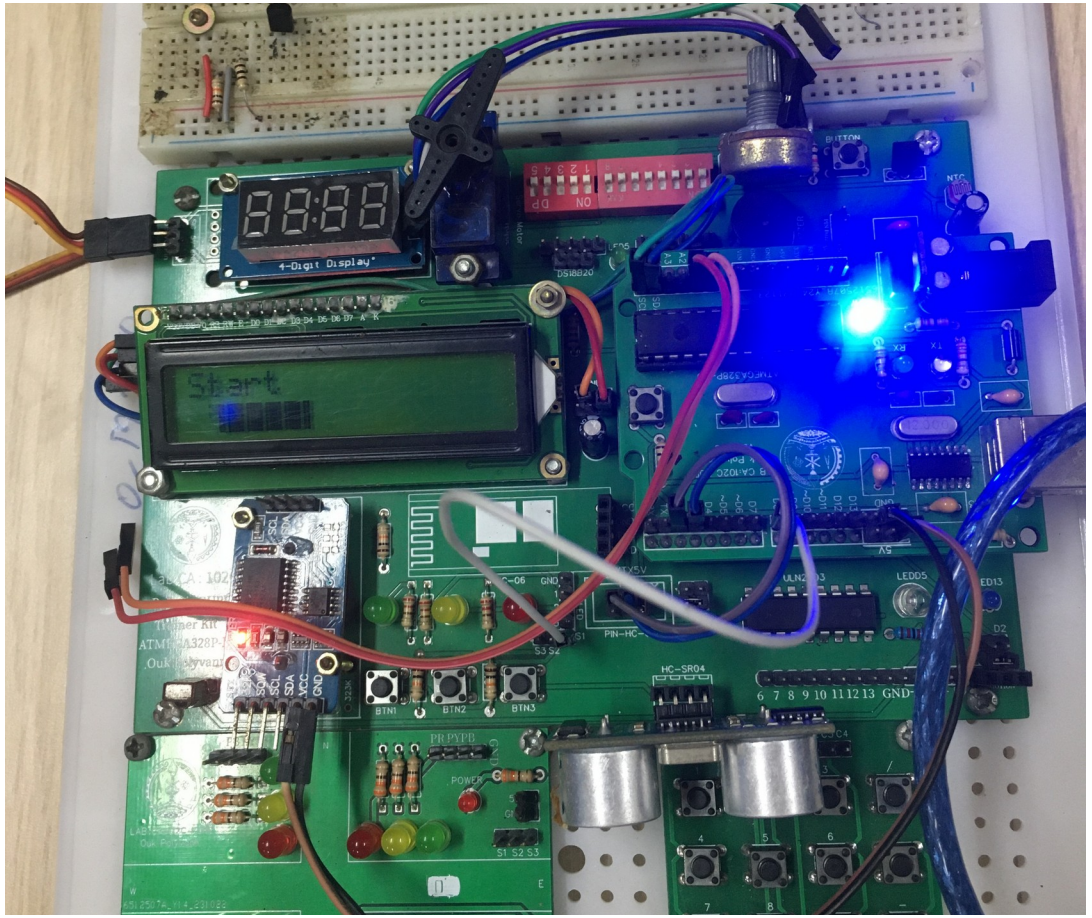
-LCD



-Momentary button or Switch

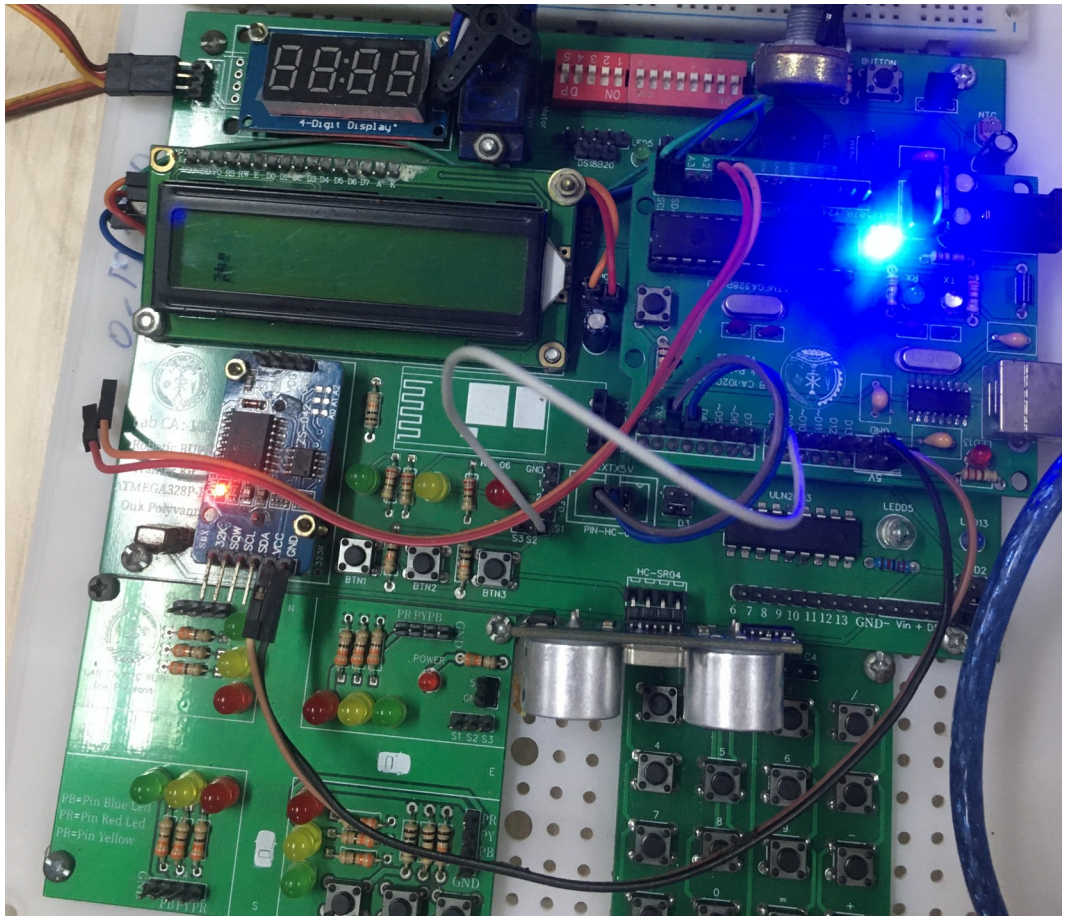
-hook-up wires

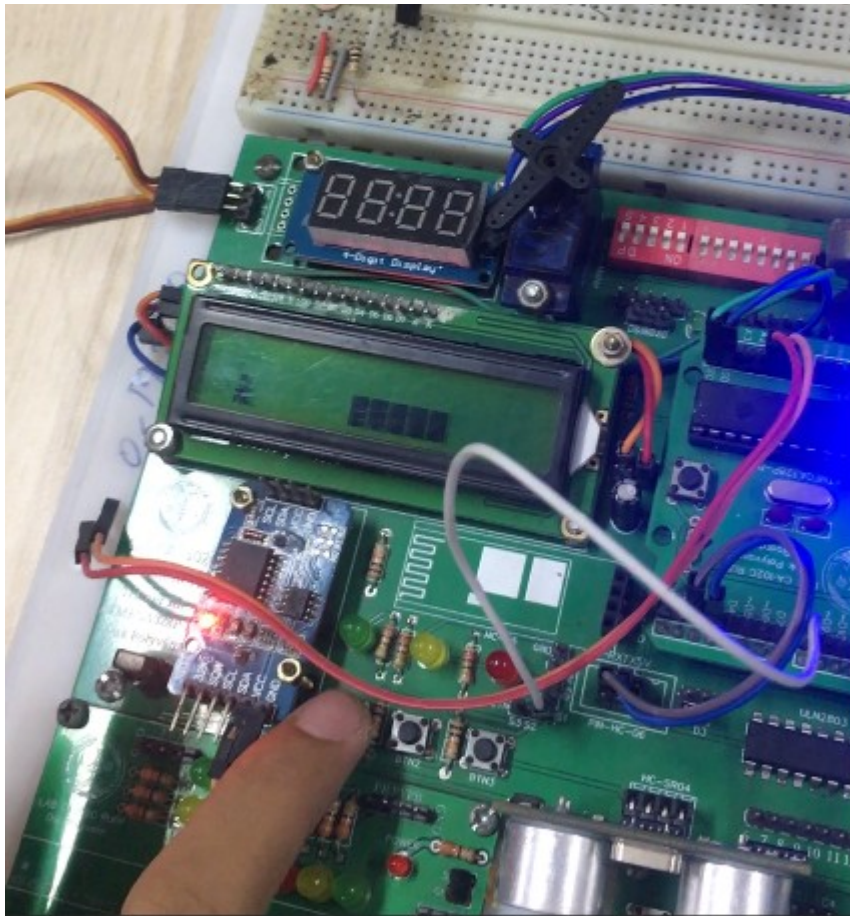
ណែនាំអំពី GAME



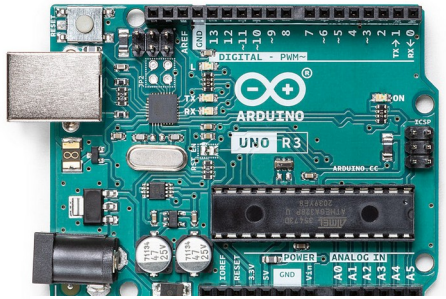
- Arduino LCD Run Game គឺជាGameដែលតម្រូវឲ្យលោតផ្លោះរបង។
- ដើម្បីលេងGameមួយនេះអ្នកត្រូវចុចButton
- កាលណាអ្នកប៉ះរបង នោះ ត្រូវបានចុះចាញ់ ហើយត្រូវបានបញ្ចប់។
- ចុចButtonជាប់គឺលោត ប្រសិនបើគឺចុះមកវិញ
- អ្នកអាចលេងវាសារជាថ្មីបានដោយគ្រាន់តែចុចButton នៅៈGameនឹងចាប់ផ្តើមម្តងទៀត។

រូបភាពដំណើរការនៃGAME





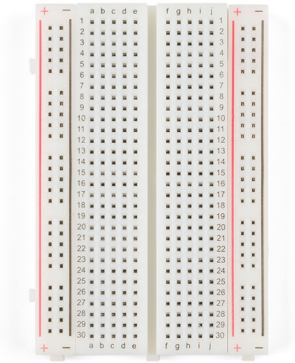
Hardware



Arduino Uno គឺជាបន្ទះមីក្រូត្រួតពិនិត្យប្រភពបើកចំហដែលមានមូលដ្ឋានលើ [illegible]28P microcontroller (MCU) និងត្រូវបានបង្កើតឡើងដោយ Arduino.cc ហើយត្រូវបានចេញផ្សាយដំបូងក្នុងឆ្នាំ 2010 ។

LCD 16 x2 គឺជាប្រភេទអេក្រង់គ្រីស្តាល់រាវ (□□□) ដែលអាចបង្ហាញដល់ទៅ 16 តួអក្សរក្នុងមួយបន្ទាត់ និង 2 បន្ទាត់។ ការបង្ហាញទាំងនេះត្រូវបានប្រើប្រាស់យ៉ាងទូលំទូលាយនៅក្នុងកម្មវិធីផ្សេងៗ ដូចជាការបង្ហាញអត្ថបទ ឬទិន្នន័យនៅក្នុងគម្រោងអេឡិចត្រូនិក។

ប៊ូតុងរុញ ឬប៊ូតុងសាមញ្ញ គឺជាយន្តការប្តូរដ៏សាមញ្ញមួយ ដើម្បីគ្រប់គ្រងទិដ្ឋភាពមួយចំនួននៃម៉ាស៊ីន ឬដំណើរការ។ ប៊ូតុងជាធម្មតាត្រូវបានផលិតចេញពីវត្ថុរឹង ជាធម្មតាប្លាស្ទិក ឬដែក។



breadboardគឺជាមូលដ្ឋានសំណង់ដែលត្រូវបានប្រើដើម្បីកសាងគំរូពាក់កណ្តាលអចិន្ត្រៃយ៍នៃសៀគ្វីអេឡិចត្រូនិ។ មិនដូច perfboard ឬ stripboard ទេ breadboards មិនតម្រូវឱ្យមានការ soldering ឬការបំផ្លាញផ្លូវហើយដូច្នេះគឺអាចប្រើឡើងវិញបាន។ សម្រាប់ហេតុផលនេះ ក្តារខៀនក៏មានប្រជាប្រិយភាពជាមួយសិស្សានុសិស្ស និងក្នុងការអប់រំផ្នែកបច្ចេកវិទ្យាផងដែរ។

Hook-Up Wire គឺជាខ្សែដែលមានអ៊ីសូឡង់តែមួយ ដែលអាចប្រើសម្រាប់កម្មវិធីដែលមានតង់ស្យុងទាប និងចរន្តទាប។ ខ្សែ Hook-Up ដំណើរការបានយ៉ាងល្អនៅក្នុងកន្លែងបិទជិតហើយភ្ជាប់មកជាមួយនូវសំណង់ជាច្រើនប្រភេទដែលមានចំហាយទឹក អ៊ីសូឡង់។

CODE

```
#include <LiquidCrystal_I2C.h>
#define SPRITE_RUN1 1
#define SPRITE_RUN2 2
#define SPRITE_JUMP 3
#define SPRITE_JUMP_UPPER '.' // Use the '.' character for the head
#define SPRITE_JUMP_LOWER 4
#define SPRITE_TERRAIN_EMPTY ' ' // User the ' ' character
#define SPRITE_TERRAIN_SOLID 5
#define SPRITE_TERRAIN_SOLID_RIGHT 6
#define SPRITE_TERRAIN_SOLID_LEFT 7
#define HERO_HORIZONTAL_POSITION 1 // Horizontal position of hero on screen
#define TERRAIN_WIDTH 16
#define TERRAIN_EMPTY 0
#define TERRAIN_LOWER_BLOCK 1
#define TERRAIN_UPPER_BLOCK 2
#define HERO_POSITION_OFF 0          // Hero is invisible
#define HERO_POSITION_RUN_LOWER_1 1  // Hero is running on lower row (pose 1)
#define HERO_POSITION_RUN_LOWER_2 2  //                               (pose 2)
#define HERO_POSITION_JUMP_1 3       // Starting a jump
#define HERO_POSITION_JUMP_2 4       // Half-way up
#define HERO_POSITION_JUMP_3 5       // Jump is on upper row
#define HERO_POSITION_JUMP_4 6       // Jump is on upper row
#define HERO_POSITION_JUMP_5 7       // Jump is on upper row
#define HERO_POSITION_JUMP_6 8       // Jump is on upper row
#define HERO_POSITION_JUMP_7 9       // Half-way down
#define HERO_POSITION_JUMP_8 10      // About to land
#define HERO_POSITION_RUN_UPPER_1 11 // Hero is running on upper row (pose 1)
#define HERO_POSITION_RUN_UPPER_2 12 //                               (pose 2)
// LiquidCrystal_I2C lcd(80, 16, 2);
LiquidCrystal_I2C lcd(0x27, 16, 2);
static char terrainUpper[TERRAIN_WIDTH + 1];
```

```
static char terrainLower[TERRAIN_WIDTH + 1];
static bool buttonPushed = false;
static int melodyPin = 3;
int ledpin = 13;
int btnpin = 8;

void initializeGraphics()
{
  static byte graphics[] =
      {
          // Run position 1
          B01100,
          B01100,
          B00000,
          B01110,
          B11100,
          B01100,
          B11010,
          B10011,
          // Run position 2
          B01100,
          B01100,
          B00000,
          B01100,
          B01100,
          B01100,
          B01100,
          B01110,
          // Jump
```

```
            B00011,
            B00011,
            B00011,
            B00011,
            B00011,
            B00011,
            // Ground left
            B11000,
            B11000,
            B11000,
            B11000,
            B11000,
            B11000,
            B11000,
            B11000,
        };
    int i;
    // Skip using character 0, this allows lcd.print() to be used to quickly draw multiple characters
    for (i = 0; i < 7; ++i)
    {
      lcd.createChar(i + 1, &graphics[i * 8]);
    }
    for (i = 0; i < TERRAIN_WIDTH; ++i)
    {
      terrainUpper[i] = SPRITE_TERRAIN_EMPTY;
      terrainLower[i] = SPRITE_TERRAIN_EMPTY;
    }
}

            B01100,
            B01100,
            B00000,
            B11110,
            B01101,
            B11111,
            B10000,
            B00000,
            // Jump lower
            B11110,
            B01101,
            B11111,
            B10000,
            B00000,
            B00000,
            B00000,
            B00000,
            // Ground
            B11111,
            B11111,
            B11111,
            B11111,
            B11111,
            B11111,
            B11111,
            B11111,
            // Ground right
            B00011,
            B00011,
```

```
119  void setup()
120  {
121    lcd.init();
122    initializeGraphics();
123
124    pinMode(ledpin, OUTPUT);
125    pinMode(melodyPin, OUTPUT);
126  }
127
128  void advanceTerrain(char *terrain, byte newTerrain) // Slide the terrain to the left in half-character increments
129  {
130    for (int i = 0; i < TERRAIN_WIDTH; ++i)
131    {
132      char current = terrain[i];
133      char next = (i == TERRAIN_WIDTH - 1) ? newTerrain : terrain[i + 1];
134      switch (current)
135      {
136      case SPRITE_TERRAIN_EMPTY:
137        terrain[i] = (next == SPRITE_TERRAIN_SOLID) ? SPRITE_TERRAIN_SOLID_RIGHT : SPRITE_TERRAIN_EMPTY;
138        break;
139      case SPRITE_TERRAIN_SOLID:
140        terrain[i] = (next == SPRITE_TERRAIN_EMPTY) ? SPRITE_TERRAIN_SOLID_LEFT : SPRITE_TERRAIN_SOLID;
141        break;
142      case SPRITE_TERRAIN_SOLID_RIGHT:
143        terrain[i] = SPRITE_TERRAIN_SOLID;
144        break;
145      case SPRITE_TERRAIN_SOLID_LEFT:
146        terrain[i] = SPRITE_TERRAIN_EMPTY;
147        break;
```

```
147         break;
148       }
149     }
150   }
151
152   bool drawHero(byte position, char *terrainUpper, char *terrainLower, unsigned int score)
153   {
154     bool collide = false;
155     char upperSave = terrainUpper[HERO_HORIZONTAL_POSITION];
156     char lowerSave = terrainLower[HERO_HORIZONTAL_POSITION];
157     byte upper, lower;
158     switch (position)
159     {
160     case HERO_POSITION_OFF:
161       upper = lower = SPRITE_TERRAIN_EMPTY;
162       break;
163     case HERO_POSITION_RUN_LOWER_1:
164       upper = SPRITE_TERRAIN_EMPTY;
165       lower = SPRITE_RUN1;
166       break;
167     case HERO_POSITION_RUN_LOWER_2:
168       upper = SPRITE_TERRAIN_EMPTY;
169       lower = SPRITE_RUN2;
170       break;
171     case HERO_POSITION_JUMP_1:
172     case HERO_POSITION_JUMP_8:
173       upper = SPRITE_TERRAIN_EMPTY;
174       lower = SPRITE_JUMP;
175       break;
176     case HERO_POSITION_JUMP_2:
```

```
177     case HERO_POSITION_JUMP_7:
178       upper = SPRITE_JUMP_UPPER;
179       lower = SPRITE_JUMP_LOWER;
180       break;
181     case HERO_POSITION_JUMP_3:
182     case HERO_POSITION_JUMP_4:
183     case HERO_POSITION_JUMP_5:
184     case HERO_POSITION_JUMP_6:
185       upper = SPRITE_JUMP;
186       lower = SPRITE_TERRAIN_EMPTY;
187       break;
188     case HERO_POSITION_RUN_UPPER_1:
189       upper = SPRITE_RUN1;
190       lower = SPRITE_TERRAIN_EMPTY;
191       break;
192     case HERO_POSITION_RUN_UPPER_2:
193       upper = SPRITE_RUN2;
194       lower = SPRITE_TERRAIN_EMPTY;
195       break;
196     }
197     if (upper != ' ')
198     {
199       terrainUpper[HERO_HORIZONTAL_POSITION] = upper;
200       collide = (upperSave == SPRITE_TERRAIN_EMPTY) ? false : true;
201     }
202     if (lower != ' ')
203     {
204       terrainLower[HERO_HORIZONTAL_POSITION] = lower;
205       collide |= (lowerSave == SPRITE_TERRAIN_EMPTY) ? false : true;
206     }
207     byte digits = (score > 9999) ? 5 : (score > 999) ? 4
208                                      : (score > 99)    ? 3
209                                      : (score > 9)     ? 2
210                                                        : 1;
211     // Draw the scene
212     terrainUpper[TERRAIN_WIDTH] = '.';
213     terrainLower[TERRAIN_WIDTH] = '.';
214     char temp = terrainUpper[16 - digits];
215     terrainUpper[16 - digits] = '.';
216     lcd.setCursor(0, 0);
217     lcd.print(terrainUpper);
218     terrainUpper[16 - digits] = temp;
219     lcd.setCursor(0, 1);
220     lcd.print(terrainLower);
221     lcd.setCursor(16 - digits, 0);
222     lcd.print(score);
223     terrainUpper[HERO_HORIZONTAL_POSITION] = upperSave;
224     terrainLower[HERO_HORIZONTAL_POSITION] = lowerSave;
225     return collide;
226   }
227
228   void buttonPush() // Handle the button push.
229   {
230     buttonPushed = true;
231   }
232
233   void loop()
234   {
```

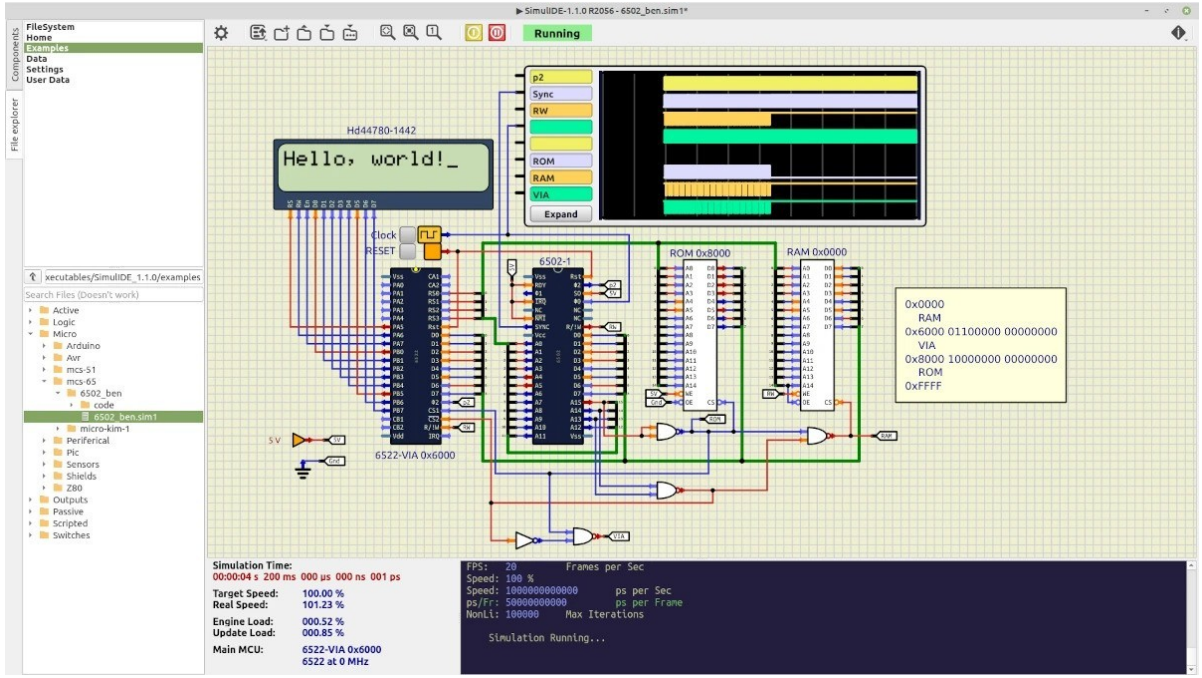
```
264   advanceTerrain(terrainUpper, newTerrainType == TERRAIN_UPPER_BLOCK ? SPRITE_TERRAIN_SOLID : SPRITE_TERRAIN_EMPTY);
265   // Make new terrain to enter on the right
266   if (--newTerrainDuration == 0)
267   {
268     if (newTerrainType == TERRAIN_EMPTY)
269     {
270       newTerrainType = (random(3) == 0) ? TERRAIN_UPPER_BLOCK : TERRAIN_LOWER_BLOCK;
271       newTerrainDuration = 2 + random(10);
272     }
273     else
274     {
275       newTerrainType = TERRAIN_EMPTY;
276       newTerrainDuration = 10 + random(10);
277     }
278   }
279   if (buttonPushed)
280   {
281     if (heroPos <= HERO_POSITION_RUN_LOWER_2)
282       heroPos = HERO_POSITION_JUMP_1;
283     buttonPushed = false;
284   }
285   if (drawHero(heroPos, terrainUpper, terrainLower, distance >> 3))
286   {
287     playing = false; // The hero collided with something. Too bad.
288     noTone(melodyPin);
289     tone(melodyPin, 200, 200); // PORT, NOTE, TIME.
290   }
291   else
292   {
```

```
235   buttonCheck();
236   static byte heroPos = HERO_POSITION_RUN_LOWER_1;
237   static byte newTerrainType = TERRAIN_EMPTY;
238   static byte newTerrainDuration = 1;
239   static bool playing = false;
240   static bool blink = false;
241   static unsigned int distance = 0;
242   if (!playing)
243   {
244     drawHero((blink) ? HERO_POSITION_OFF : heroPos, terrainUpper, terrainLower, distance >> 3);
245     if (blink)
246     {
247       lcd.setCursor(0, 0);
248       lcd.print("Start");
249     }
250     delay(250);
251     blink = !blink;
252     if (buttonPushed)
253     {
254       initializeGraphics();
255       heroPos = HERO_POSITION_RUN_LOWER_1;
256       playing = true;
257       buttonPushed = false;
258       distance = 0;
259     }
260     return;
261   }
262   // Shift the terrain to the left
263   advanceTerrain(terrainLower, newTerrainType == TERRAIN_LOWER_BLOCK ? SPRITE_TERRAIN_SOLID : SPRITE_TERRAIN_EMPTY);
```

```
293       if (heroPos == HERO_POSITION_RUN_LOWER_2 || heroPos == HERO_POSITION_JUMP_8)
294       {
295         heroPos = HERO_POSITION_RUN_LOWER_1;
296       }
297       else if ((heroPos >= HERO_POSITION_JUMP_3 && heroPos <= HERO_POSITION_JUMP_5) && terrainLower[HERO_HORIZONTAL_POSITION] != SPRITE
298       {
299         heroPos = HERO_POSITION_RUN_UPPER_1;
300       }
301       else if (heroPos >= HERO_POSITION_RUN_UPPER_1 && terrainLower[HERO_HORIZONTAL_POSITION] == SPRITE_TERRAIN_EMPTY)
302       {
303         heroPos = HERO_POSITION_JUMP_5;
304       }
305       else if (heroPos == HERO_POSITION_RUN_UPPER_2)
306       {
307         heroPos = HERO_POSITION_RUN_UPPER_1;
308       }
309       else
310       {
311         ++heroPos;
312       }
313       ++distance;
314     }
315     delay(100);
316   }
317
318   void buttonCheck()
319   {
320     int btnstate = digitalRead(btnpin);
321     if (btnstate == HIGH)
```

```
309       else
310       {
311         ++heroPos;
312       }
313       ++distance;
314     }
315     delay(100);
316   }
317
318   void buttonCheck()
319   {
320     int btnstate = digitalRead(btnpin);
321     if (btnstate == HIGH)
322     {
323       buttonPushed = true;
324       digitalWrite(ledpin, HIGH);
325     }
326     else
327     {
328       digitalWrite(ledpin, LOW);
329     }
330   }
```

Simulation



- SimulIDE is a simple real time electronic circuit simulator, intended for hobbyist or students to learn and experiment with analog and digital electronic circuits and microcontrollers.
- It supports PIC, AVR , Arduino and other MCUs and MPUs
- Thus, We use SimulIDE to simulate our ARDUINO LCD Game.

Circuit of Arduino LCD GAME