ME 327: Design and Control of Haptic Systems
Spring 2020

Lecture 14:
Hapkit Programming

Allison M. Okamura
Stanford University

Hapkit Board

Hapkit board "front"

reset button

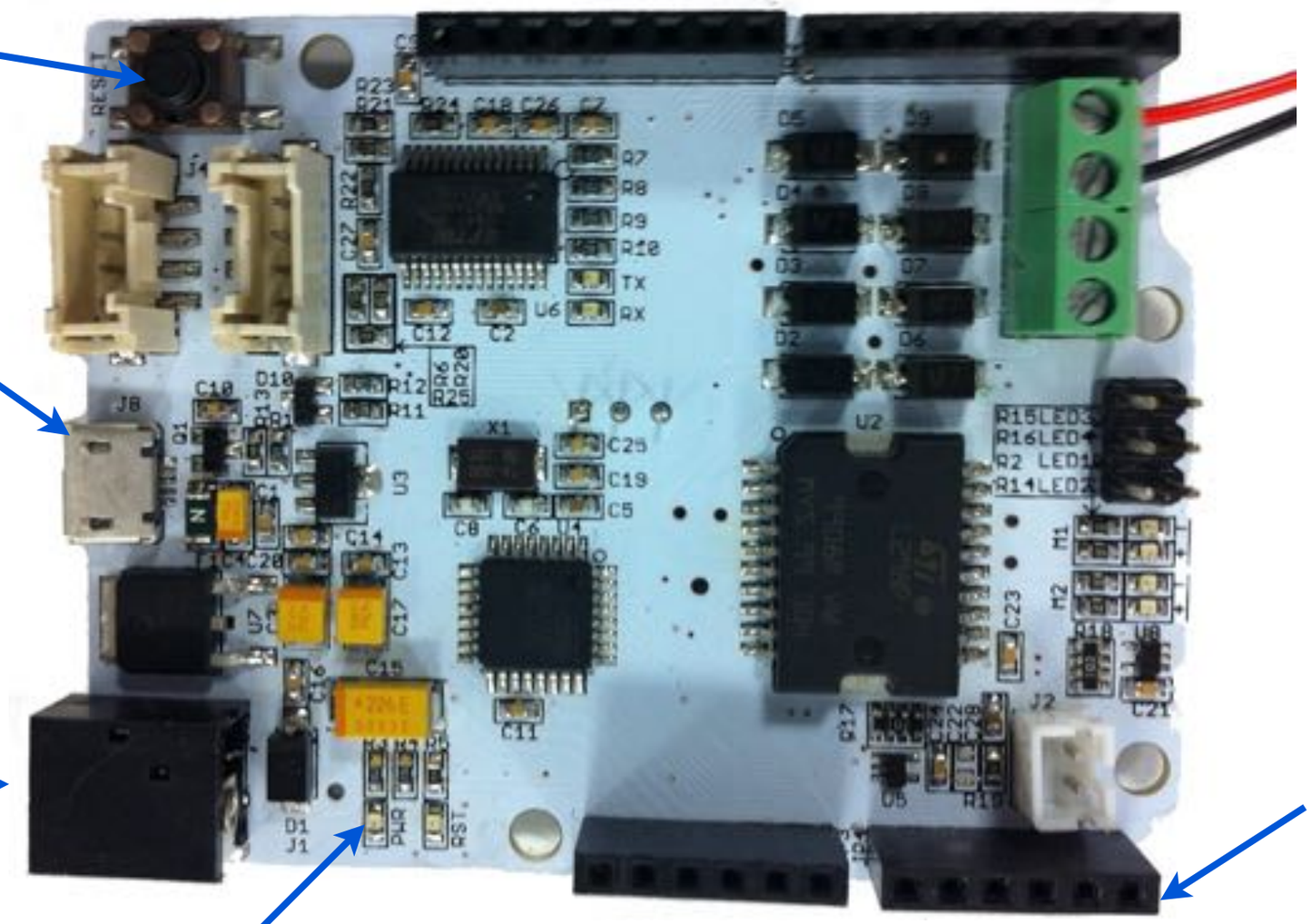
Micro USB connector

connector for power adaptor (for motor)

power LED

motor leads

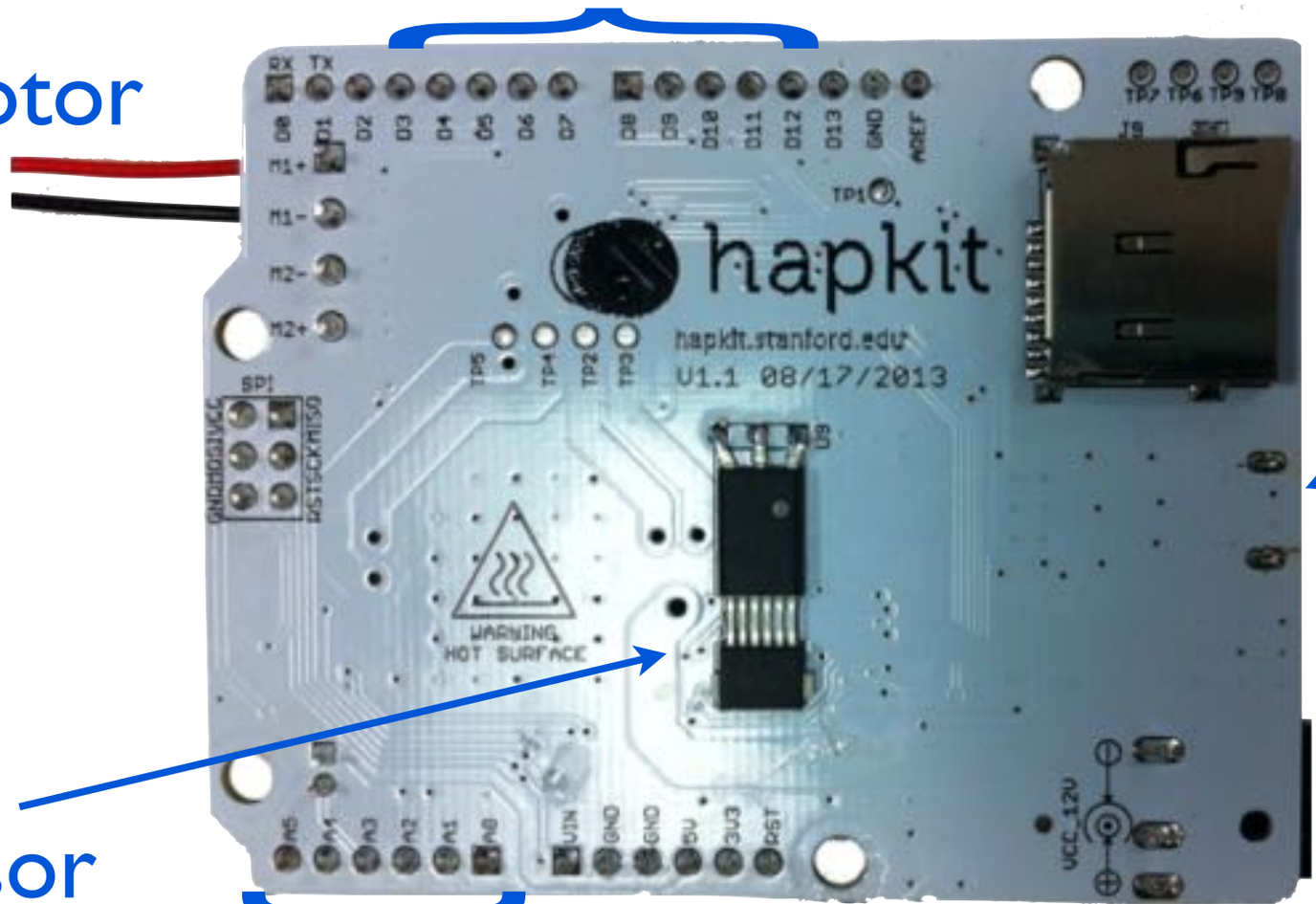
pins available for read/ write



Hapkit board "back"

digital input/output pins

to motor



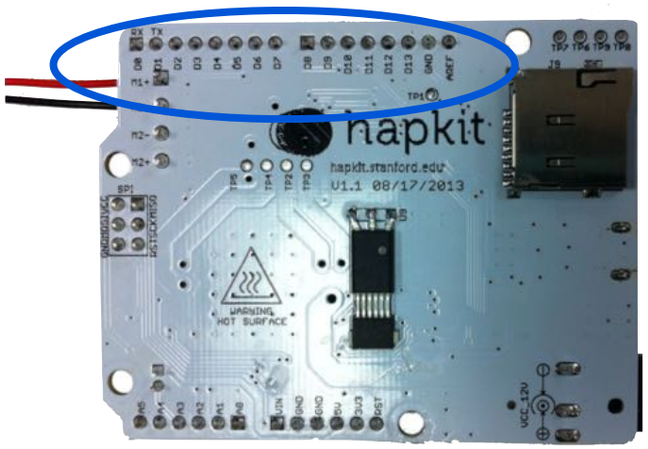
Micro USB connector

connector for power adaptor (for motor)

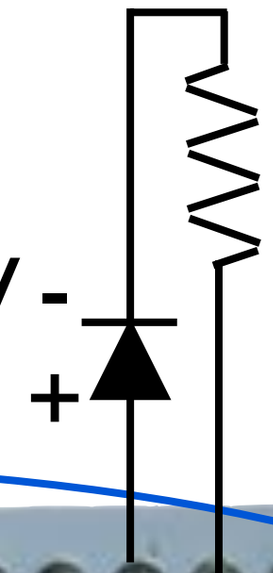
MR sensor

analog input pins

LED connection

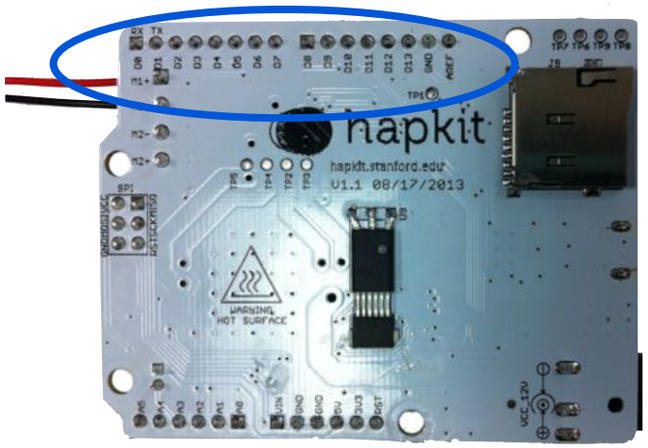


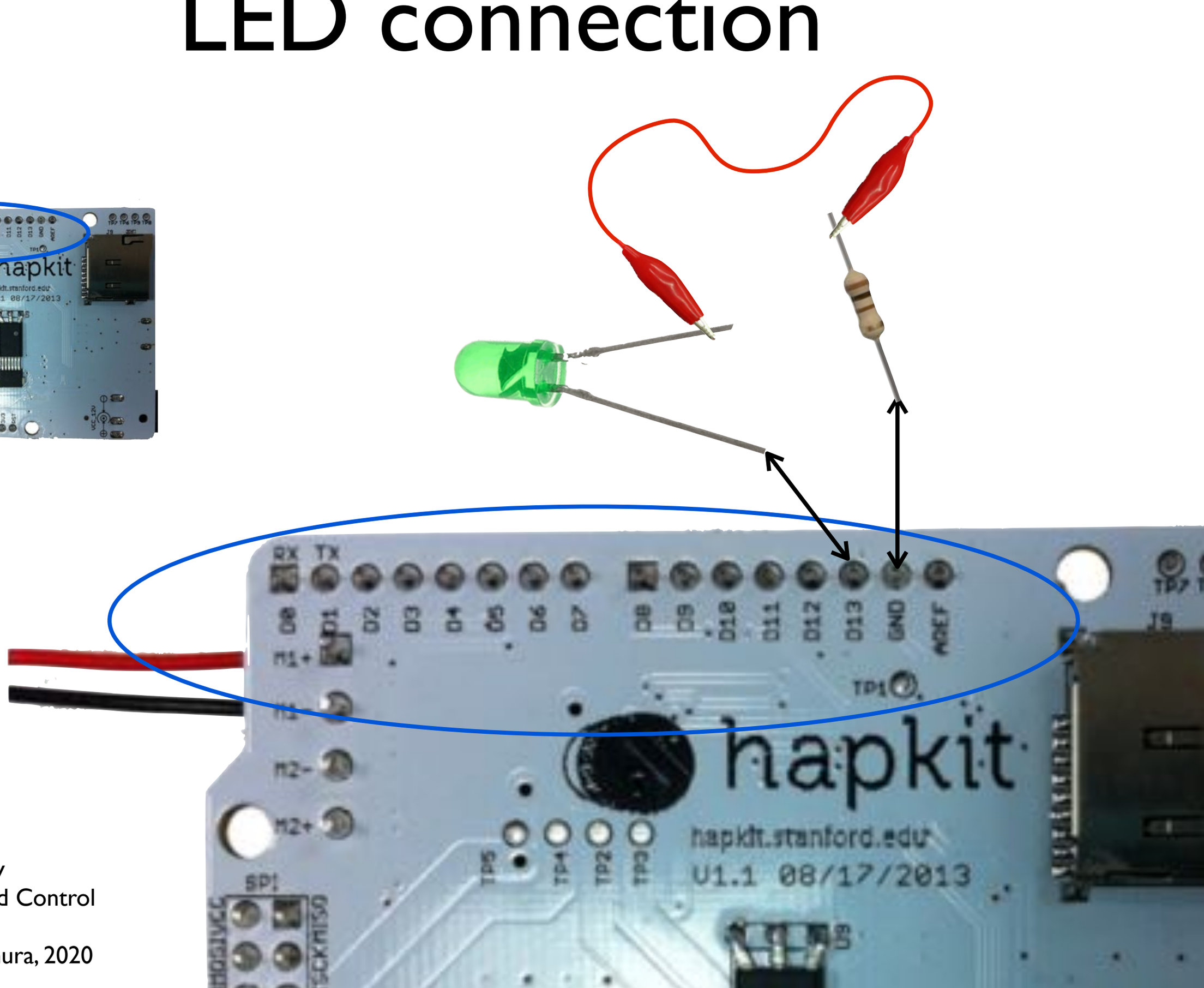
When commanded to go HIGH, pin 13 will output approximately 5 V -



+

LED connection





Hapkit Programming Basics

Programming Hapkit Board

Arduino is a single-board microcontroller that makes using electronics in multidisciplinary projects relatively easy

The Hapkit board is based on the Arduino design, with some added features

The Hapkit board can be programmed using the same Arduino integrated development environment (IDE) as an Arduino board

Follow the instructions in the assignment to download, install, and test the Arduino software

Example Hapkit Program

Programming Area

Code:

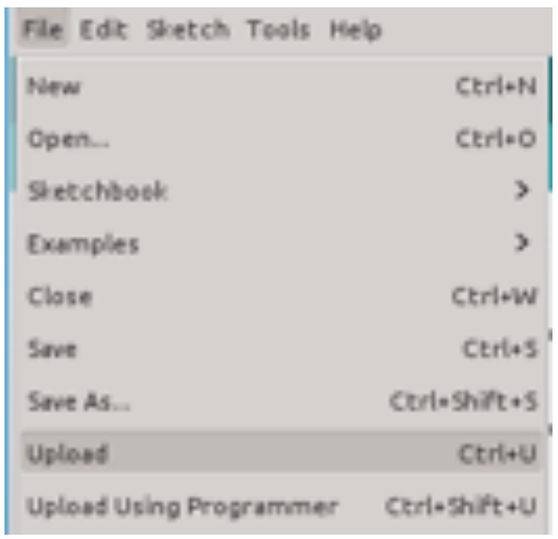
```
// Blink the pin 13 LED
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

The Arduino programming language is similar to C and java

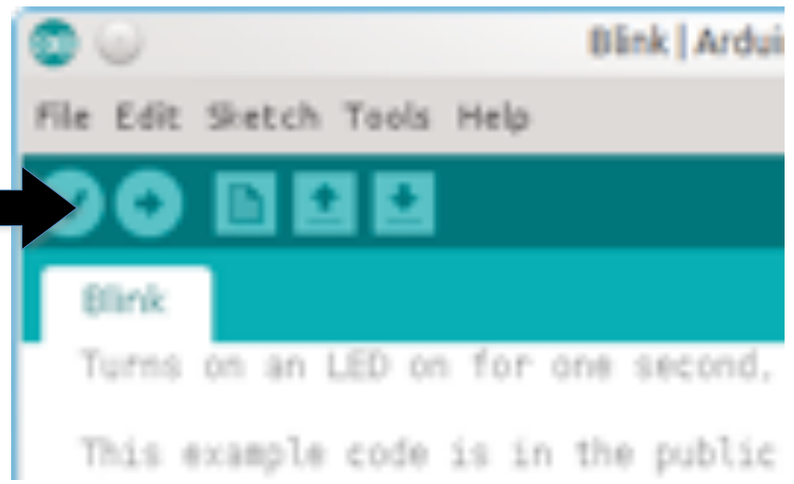
material courtesy Paulo Blikstein

Running a Hapkit Program



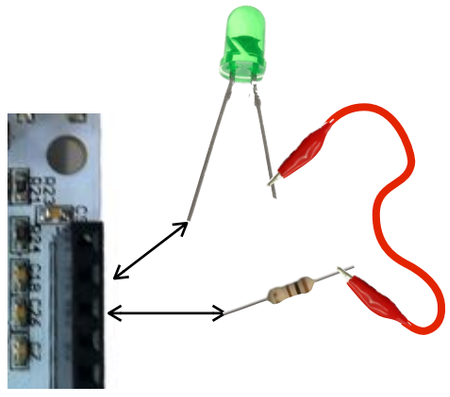
Click File > Upload

or click the Upload button



Clicking Upload sends the code from the computer to the Hapkit board

Your code is now running on the Hapkit Board! What is the LED doing?



material courtesy Paulo Blikstein

Understanding the code

```
// Blink the pin 13 LED
void setup(){
  pinMode(13, OUTPUT);
}

void loop(){
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

void setup() { … }
Any code inside the setup function runs *once* to setup the Arduino

void loop() { … }
Any code inside the loop function loops over and over again

material courtesy Paulo Blikstein

The Serial Monitor

- The Arduino environment's built-in serial monitor can be used to communicate with your Hapkit board
- In the menu, click on **Tools > Serial Monitor** to view
- Use the following code to interact with the serial monitor:

```
void setup() {
  Serial.begin(9600);  // open the serial port at 9600 bps
}
```

to print to the serial monitor:

```
Serial.println("Hello World!");

int my_variable = 0; // variable
Serial.println(my_variable);
```

to read from the serial monitor:

```
if (Serial.available() > 0)
{
  char inByte = Serial.read();
}
```


Programming syntax/hints

- Use // before any text you want to have as a comment (and comment well!)
- Each statement must end with a ; (semicolon)
- You must declare variables before you use them
- Call built-in Arduino functions to perform I/O (input/output)
- See http://arduino.cc/en/Reference/HomePage for a language reference that describes structure, variables, and functions
- If you have never programmed before, the easiest way to learn is by looking at and modifying example programs. Don't modify too many things at once before testing your code. Many examples under File > Examples

Global variables

Hapkit_Template1 | Arduino 1.8.12

Hapkit_Template1

```
//------------------------------------------------------------------------
// Code to test basic Hapkit functionality (sensing and force output)
// Updated by Allison Okamura 1.17.2018
//------------------------------------------------------------------------

// Includes
#include <math.h>

// Pin declares
int pwmPin = 5; // PWM output pin for motor 1
int dirPin = 8; // direction output pin for motor 1
int sensorPosPin = A2; // input pin for MR sensor
int fsrPin = A3; // input pin for FSR sensor

// Position tracking variables
int updatedPos = 0;     // keeps track of the latest updated value of the MR sensor reading
int rawPos = 0;         // current raw reading from MR sensor
int lastRawPos = 0;     // last raw reading from MR sensor
int lastLastRawPos = 0; // last last raw reading from MR sensor
int flipNumber = 0;     // keeps track of the number of flips over the 180deg mark
int tempOffset = 0;
int rawDiff = 0;
int lastRawDiff = 0;
int rawOffset = 0;
int lastRawOffset = 0;
const int flipThresh = 700;  // threshold to determine whether or not a flip over the 180 degree mark occurred
boolean flipped = false;
double OFFSET = 980;
double OFFSET_NEG = 15;

// Kinematics variables
double xh = 0;           // position of the handle [m]

// Force output variables
double force = 0;           // force at the handle
double Tp = 0;              // torque of the motor pulley
double duty = 0;            // duty cylce (between 0 and 255)
unsigned int output = 0;    // output command to the motor
```

```
/Users/allisono/Google Drive/Teaching/ME327 Spring 2020/Hapkit/Hapkit code/2019 code/Hapkit_Template1/Hapkit_Templat
Invalid library found in /Users/allisono/Documents/Arduino/libraries/pinchangeint: no headers files (.h) found in /U
```

1 Arduino Uno on /dev/cu.usbserial-AI03FMVX

Setup function

Hapkit_Template1 | Arduino 1.8.12

Hapkit_Template1

```
/ ------------------------------------------------------------------
/ Setup function -- NO NEED TO EDIT
/ ------------------------------------------------------------------
oid setup()

 // Set up serial communication
 Serial.begin(9600);

 // Set PWM frequency
 setPwmFrequency(pwmPin,1);

 // Input pins
 pinMode(sensorPosPin, INPUT); // set MR sensor pin to be an input
 pinMode(fsrPin, INPUT);       // set FSR sensor pin to be an input

 // Output pins
 pinMode(pwmPin, OUTPUT);  // PWM pin for motor A
 pinMode(dirPin, OUTPUT);  // dir pin for motor A

 // Initialize motor
 analogWrite(pwmPin, 0);     // set to not be spinning (0/255)
 digitalWrite(dirPin, LOW);  // set direction

 // Initialize position valiables
 lastLastRawPos = analogRead(sensorPosPin);
 lastRawPos = analogRead(sensorPosPin);
 flipNumber = 0;
```

```
/Users/allisono/Google Drive/Teaching/ME327 Spring 2020/Hapkit/Hapkit code/2019 code/Hapkit_Template1/Hapkit_Templa
Invalid library found in /Users/allisono/Documents/Arduino/libraries/pinchangeint: no headers files (.h) found in /
```

1 Arduino Uno on /dev/cu.usbserial-AL00BLMX

Main Loop

Hapkit_Template1 | Arduino 1.8.12

Hapkit_Template1

```
// ------------------------------------------------------------------
// Main Loop
// ------------------------------------------------------------------
void loop()
{
 //*****************************************************************
 //*** Section 1. Compute position in counts (do not change) ***
 //*****************************************************************

 // Get voltage output by MR sensor
 rawPos = analogRead(sensorPosPin);  //current raw position from MR sensor

 // Calculate differences between subsequent MR sensor readings
 rawDiff = rawPos - lastRawPos;          //difference btwn current raw positio
 lastRawDiff = rawPos - lastLastRawPos;  //difference btwn current raw positio
 rawOffset = abs(rawDiff);
 lastRawOffset = abs(lastRawDiff);

 // Update position record-keeping vairables
 lastLastRawPos = lastRawPos;
 lastRawPos = rawPos;

 // Keep track of flips over 180 degrees
 if((lastRawOffset > flipThresh) && (!flipped)) { // enter this anytime the la
   if(lastRawDiff > 0) {        // check to see which direction the drive whee
     flipNumber--;              // cw rotation
   } else {                     // if(rawDiff < 0)
     flipNumber++;              // ccw rotation
   }
   flipped = true;            // set boolean so that the next time through the
 } else {                       // anytime no flip has occurred
   flipped = false;
 }
  updatedPos = rawPos + flipNumber*OFFSET; // need to update pos based on wha
```

```
/Users/allisono/Google Drive/Teaching/ME327 Spring 2020/Hapkit/Hapkit code/2019 c
Invalid library found in /Users/allisono/Documents/Arduino/libraries/pinchangeint
```

1 Arduino Uno on /dev/cu.usbserial-AL008LMX

Hapkit_Template1 | Arduino 1.8.12

Hapkit_Template1

```
//*****************************************************************
//*** Section 2. Compute position in meters ********************
//*****************************************************************

// ADD YOUR CODE HERE
// Define kinematic parameters you may need
   //double rh = ?;   //[m]
// Step 8.1: print updatedPos via serial monitor
// Step 8.6: double ts = ?; // Compute the angle of the sector pulley (ts) in d
// Step 8.7: xh = ?;        // Compute the position of the handle (in meters) ba
// Step 8.8: print xh via serial monitor

//*****************************************************************
//*** Section 3. Assign a motor output force in Newtons ********
//*****************************************************************

// ADD YOUR CODE HERE
// Define kinematic parameters you may need
   //double rp = ?;   //[m]
   //double rs = ?;   //[m]
// Step C.1: force = ?; // You can  generate a force by assigning this to a var
// Step C.2: Tp = ?;    // Compute the require motor pulley torque (Tp) to gene

//*****************************************************************
//*** Section 4. Force output (do not change) ******************
//*****************************************************************

// Determine correct direction for motor torque
if(force > 0) {
  digitalWrite(dirPin, HIGH);
} else {
  digitalWrite(dirPin, LOW);
}

// Compute the duty cycle required to generate Tp (torque at the motor pulley)
duty = sqrt(abs(Tp)/0.0183);

// Make sure the duty cycle is between 0 and 100%
if (duty > 1) {
  duty = 1;
} else if (duty < 0) {
  duty = 0;
}
output = (int)(duty* 255);   // convert duty cycle to output signal
analogWrite(pwmPin,output);  // output the signal
}
```

```
/Users/allisono/Google Drive/Teaching/ME327 Spring 2020/Hapkit/Hapkit code/2019 c
Invalid library found in /Users/allisono/Documents/Arduino/libraries/pinchangeint:
```

PWM function

Hapkit_Template1 | Arduino 1.8.12

Hapkit_Template1

```
// ****************************************************************
// Function to set PWM Freq -- DO NOT EDIT
// ****************************************************************
void setPwmFrequency(int pin, int divisor) {
  byte mode;
  if(pin == 5 || pin == 6 || pin == 9 || pin == 10) {
    switch(divisor) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 64: mode = 0x03; break;
      case 256: mode = 0x04; break;
      case 1024: mode = 0x05; break;
      default: return;
    }
    if(pin == 5 || pin == 6) {
      TCCR0B = TCCR0B & 0b11111000 | mode;
    } else {
      TCCR1B = TCCR1B & 0b11111000 | mode;
    }
  } else if(pin == 3 || pin == 11) {
    switch(divisor) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 32: mode = 0x03; break;
      case 64: mode = 0x04; break;
      case 128: mode = 0x05; break;
      case 256: mode = 0x06; break;
      case 1024: mode = 0x7; break;
      default: return;
    }
    TCCR2B = TCCR2B & 0b11111000 | mode;
  }
}
```

```
/Users/allisono/Google Drive/Teaching/ME327 Spring 2020/Hapkit/Hapkit code/2019 code/Ha
Invalid library found in /Users/allisono/Documents/Arduino/libraries/pinchangeint: no t
```

1 Arduino Uno on /dev/cu.usbserial-AL00BLMX

Rendering on Hapkit

Environment switching

Hapkit_Template2 | Arduino 1.8.12

Hapkit_Template2

```
//----------------------------------------------------------------
// Code to render on Hapkit
// Code updated by Cara Nunez 4.17.19
//----------------------------------------------------------------
// Parameters that define what environment to render
//#define ENABLE_VIRTUAL_WALL
//#define ENABLE_LINEAR_DAMPING
//#define ENABLE_NONLINEAR_FRICTION
//#define ENABLE_HARD_SURFACE
//#define ENABLE_BUMP_VALLEY
//#define ENABLE_TEXTURE
```

Done Saving.

2 Arduino Uno on /dev/cu.usbserial-AL008LMX

Hapkit_Template2 | Arduino 1.8.12

Hapkit_Template2

```
  //*****************************************************************
  //*** Section 3. Assign a motor output force in Newtons *******
  //*****************************************************************
  //*****************************************************************
  //*****************************************************************
  //******************** Rendering Algorithms ***********************
  //*****************************************************************

#ifdef ENABLE_VIRTUAL_WALL
#endif

#ifdef ENABLE_LINEAR_DAMPING
#endif

#ifdef ENABLE_NONLINEAR_FRICTION
#endif

#ifdef ENABLE_HARD_SURFACE
#endif

#ifdef ENABLE_BUMP_VALLEY
#endif

#ifdef ENABLE_TEXTURE
#endif

  // ADD YOUR CODE HERE (Use your previous code)
  // Define kinematic parameters you may need
     //double rp = ?;   //[m]
     //double rs = ?;   //[m]
  // Step C.1: force = ?; // You can  generate a force by assigning this t
  // Step C.2: Tp = ?;    // Compute the require motor pulley torque (Tp)
```

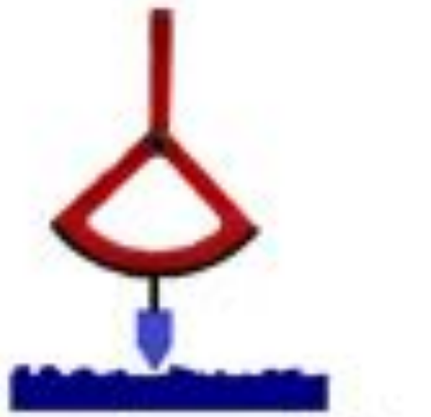
Done Saving.

180 Arduino Uno on /dev/cu.usbserial-AL008LMX

Soft Wall

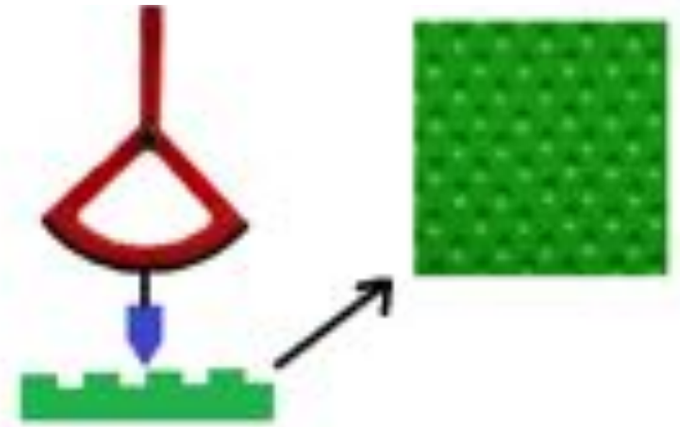
Linear Damping

Rigid Wall

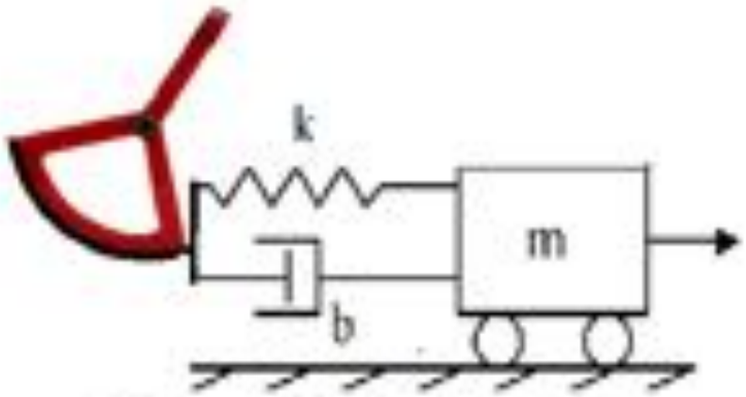


Non-Linear Friction

Bump and Valley



Textured Surface

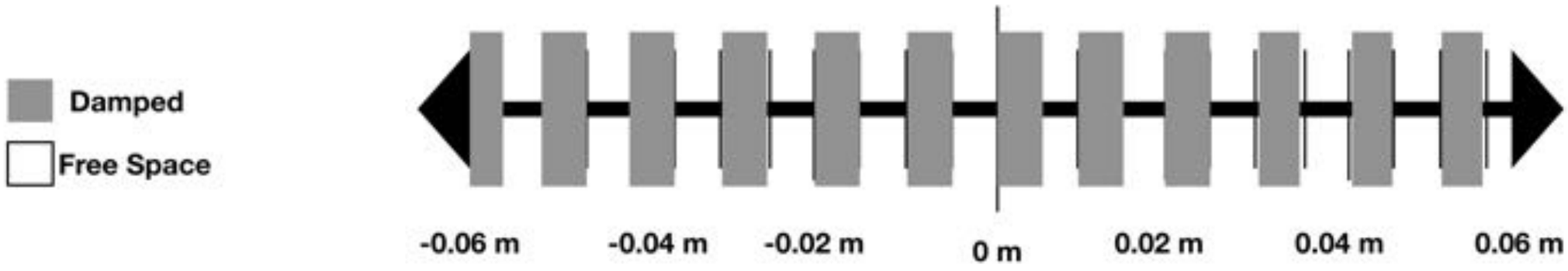


Spring-Mass System

images courtesy Tania Morimoto

Texture Example

"Given that the workspace ranges from 0.06 m to 0.06 m, I split the workspace into 24 equally sized portions. Then I added damping to half of them. Initially I had only 6 portions but needed to increase the resolution by quite a bit to make the texture more realistic."




```
#ifdef ENABLE_TEXTURE //initialize boolean

boolean inDampedArea = false;

//check if the user is within a damped area
if (xh < 0.055 || (0.05 < xh && xh < 0.045) || (0.04 < xh && xh < 0.035) || (0.03 < xh && xh <
0.025) || (0.02 < xh && xh < 0.02) || (0.015 < xh && xh < 0.01)
|| (0.005 < xh && xh < 0) || (0.005 < xh && xh < 0.01) || (0.015 < xh && xh < 0.02) || (0.025
< xh && xh < 0.03) || (0.035 < xh && xh < 0.04) || (0.045 < xh && xh < 0.05))
{
  inDampedArea = true;
}

//add damping if the user is within the damped areas
if (inDampedArea) {
  noEnvironment = false;
  double b = 5; //set damping coefficient [Ns/m]
  force = b * dxh_filt; //set force output
}

#endif
```

materials courtesy Brandon Ritter

Graphics Programming

Processing

To visualize the position of the haptic device for debugging and presentation purposes, we recommend adding graphics using the Processing language (http://processing.org)

Here are two examples to get you started:
http://www.arduino.cc/en/Tutorial/Graph
http://arduining.com/2013/08/05/arduino-and-processing-graph-example/

You need to run the Processing program in order to see the graphics.

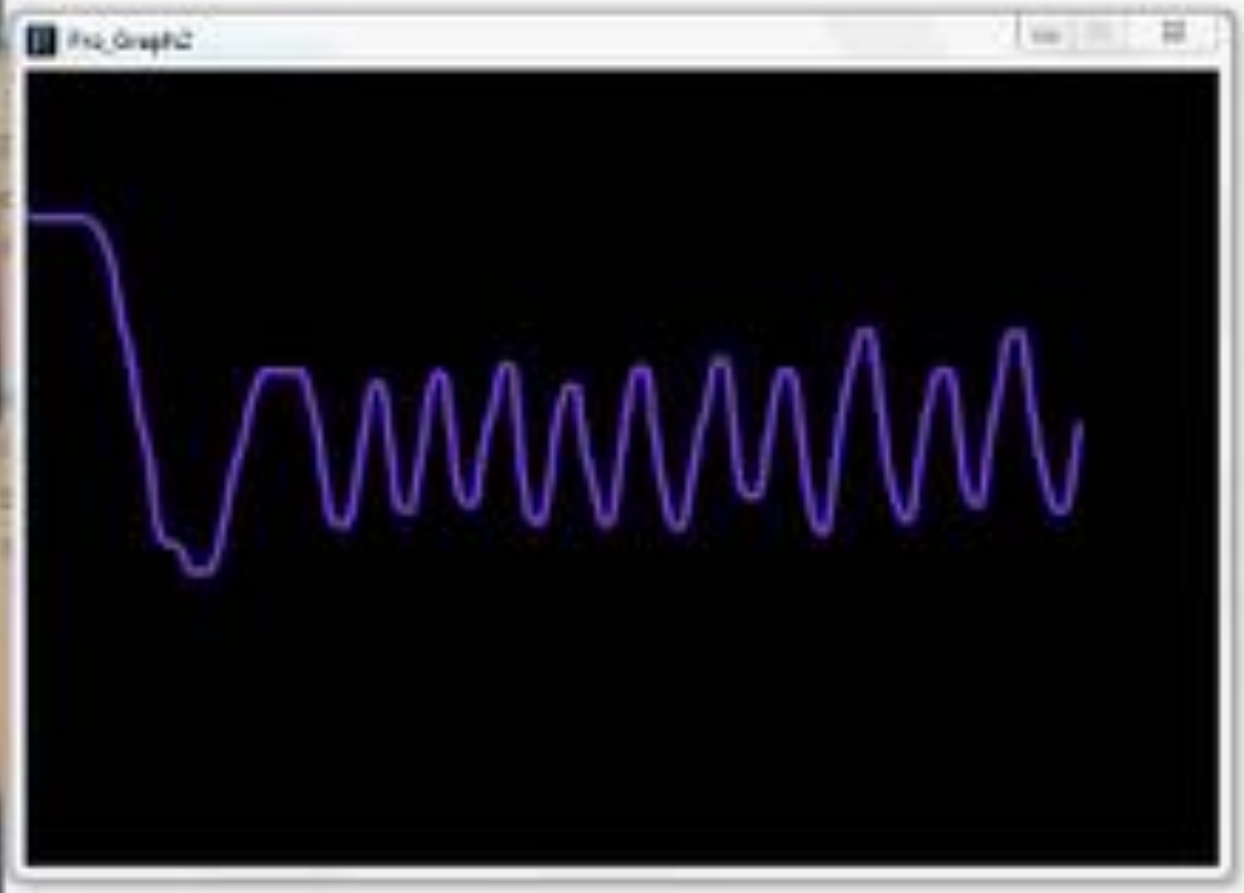
After you have the Arduino program running on your board, make sure that the USB (serial line) is connected, and run your Processing program.

The Processing program will receive the serial data and create the graphics.

Graph | Arduino 1.0.2

File Edit Sketch Tools Help

Graph




```
by Tom Igoe and Scott Fitzgerald
This example code is in the public domain.
http://www.arduino.cc/en/Tutorial/Graph
*/
```

0:37 / 0:54

Arduino and Processing (Graph Example)