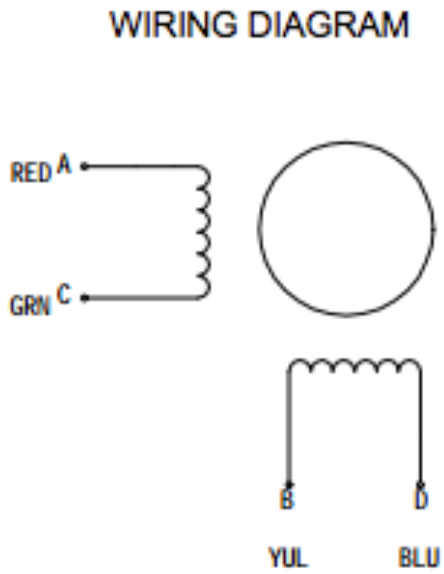
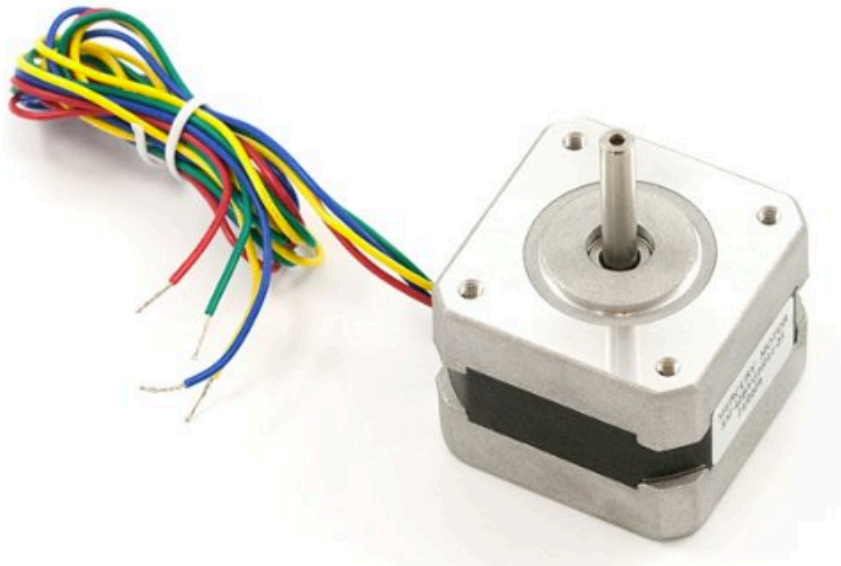
ROCO222: Intro to sensors and actuators

Lecture 5

Arduino motor shield stepper motor control

Sparksfun stepper SM-42BYG011-25

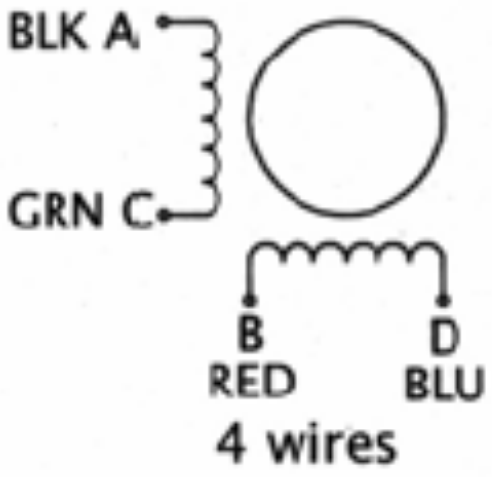




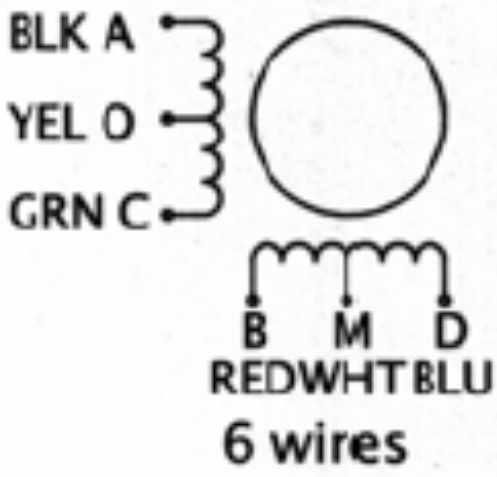
There are a number of different types of stepper motors, but in this tutorial we will specifically be addressing bipolar stepper motors. Bipolar stepper motors typically have 4 pins, which correspond to two coils. To use a stepper, you need to power these two coils in phase with alternating polarity.

Mini Hybrid Stepper Motor Size 14

Wiring Diagram



4 wires



6 wires

Connecting Stepper Motors

- For unipolar motors: to connect up the stepper, first figure out which pins connected to which coil, and which pins are the center taps.
- If its a 5-wire motor then there will be 1 that is the center tap for both coils.
- There are plenty of tutorials online on how to reverse engineer the coils pin outs.
- The center taps should both be connected together to the GND terminal on the motor shield output block, then coil 1 should connect to one motor port (say M1 or M3) and coil 2 should connect to the other motor port (M2 or M4).
- For bipolar motors: its just like unipolar motors except there's no 5th wire to connect to ground. The code is exactly the same.

Driving stepper motors

- To make a bipolar motor spin:
- Power the first coil.
- Next power the second coil with reverse polarity.
- Then power the first coil with reverse polarity.
- Finally, power the second coil.

- To reverse the motor direction of a bipolar stepper, simply reverse the polarity of the second coil.

Arduino motor shield stepper demo

- /
  ****
  ****
- Motor Shield Stepper Demo
- by Randy Sarafan

- For more information see:
- http://www.instructables.com/id/Arduino-Motor-Shield-Tutorial/

- **
  ********/

Setup()

Function	Channel A	Channel B
Direction	Digital 12	Digital 13
Speed (PWM)	Digital 3	Digital 11
Brake	Digital 9	Digital 8
Current Sensing	Analog 0	Analog 1

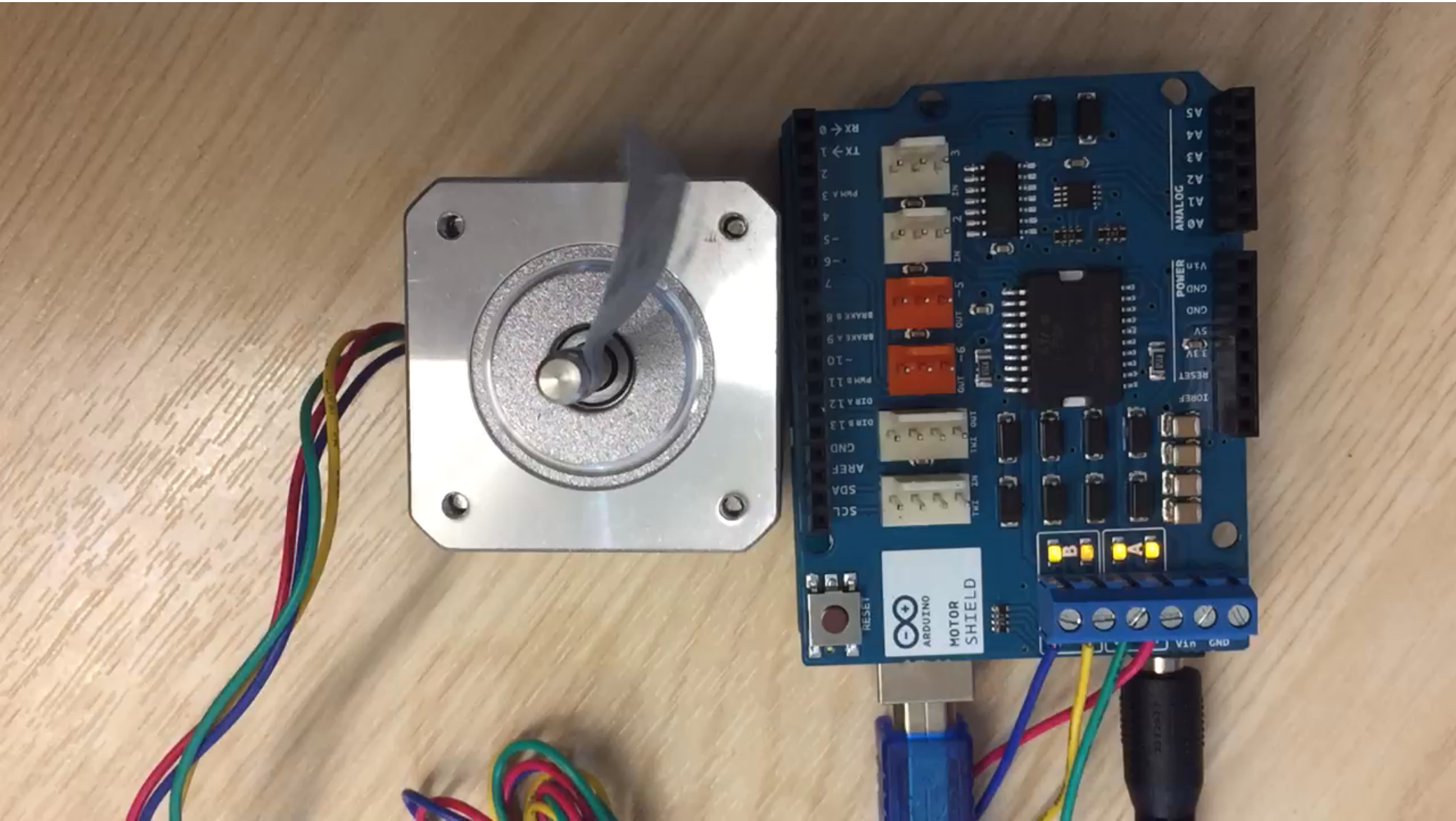
```
int delaylegnth = 30;

void setup() {

  //establish motor direction toggle pins
  pinMode(12, OUTPUT); //CH A -- HIGH = forwards and LOW = backwards???
  pinMode(13, OUTPUT); //CH B -- HIGH = forwards and LOW = backwards???

  //establish motor brake pins
  pinMode(9, OUTPUT); //brake (disable) CH A
  pinMode(8, OUTPUT); //brake (disable) CH B

}
```

loop()

```
void loop(){

  digitalWrite(9, LOW);   //ENABLE CH A
  digitalWrite(8, HIGH);  //DISABLE CH B

  digitalWrite(12, HIGH);    //Sets direction of CH A
  analogWrite(3, 255);    //Moves CH A
  delay(delaylegnth);

  digitalWrite(9, HIGH);   //DISABLE CH A
  digitalWrite(8, LOW);  //ENABLE CH B

  digitalWrite(13, LOW);    //Sets direction of CH B
  analogWrite(11, 255);    //Moves CH B
  delay(delaylegnth);

  digitalWrite(9, LOW);   //ENABLE CH A
  digitalWrite(8, HIGH);  //DISABLE CH B

  digitalWrite(12, LOW);    //Sets direction of CH A
  analogWrite(3, 255);    //Moves CH A
  delay(delaylegnth);

  digitalWrite(9, HIGH);   //DISABLE CH A
  digitalWrite(8, LOW);  //ENABLE CH B

  digitalWrite(13, HIGH);    //Sets direction of CH B
  analogWrite(11, 255);    //Moves CH B
  delay(delaylegnth);
}
```

- Power the first coil
- Next power the second coil with reverse polarity
- Then power the first coil with reverse polarity
- Finally, power the second coil.

Stepper operation



OOD Arduino code using stepper a motor shield class

```
#include "StepperMS.h"
#include "TimerOne.h"

// stepper control
// from first principles
//  Created by Ian Howard on 8/20/16.
// demo for just  fullStepSinglePhaseCCW

// Arduino motor shield connections
// Function pins per Ch.A
int Direction_A = 12;      // Direction_A         D12
int PWM_A = 3;             // PWM_A               D3
int Brake_A = 9;           // Brake_A             D9
int CurrentSensing_A = 0; // CurrentSensing_A    A0

// Function pins per Ch.B
int Direction_B = 13;      // Direction_B         D13
int PWM_B = 11;            // PWM_B               D11
int Brake_B = 8;           // Brake_B             D8
int CurrentSensing_B = 1; // CurrentSensing_B    A1

// steps for 1 revolution of stepper motor
int stepsPerRev  = 200;
```

// This is a demo cpp library class that is setup for and WILL work with the Arduino motor shield

You will need to download the TimerOne Arduino library

OOD Arduino code using stepper a motor shield class

```
//initialize an instance of Motor Sheild class
StepperMS stepper(stepsPerRev, Direction_A, PWM_A, Brake_A, Direction_B, PWM_B, Brake_B);

int timeDelay = 2;

// stepper interrupt service routine
void stepperISR()
{
  // call single phase stepper coil update
  stepper.fullStepSinglePhaseCCW();
}

void setup() {

  // setup stepper control
  stepper.Init();

  // setup timer
  Timer1.initialize(3000);
  Timer1.attachInterrupt(stepperISR);
}

void loop()
{
  // no polling functions as all interrupt driven!
}
```

// This is a demo cpp library class that is setup for and WILL work with the Arduino motor shield

CPP header for the motor shield class

```
class StepperMS {
  public:
    StepperMS(int stepsPerRev, int Direction_A, int PWM_A, int Brake_A, int Direction_B, int PWM_B, int Brake_B);
    ~StepperMS();

    // init pin connections
    void Init();
    //  stepper coil commands for different modes
    void fullStepSinglePhaseCCW();

  private:
    // control commands
    void ChanA_Forwards();
    void ChanA_Backwards();
    void ChanA_Off();
    void ChanB_Forwards();
    void ChanB_Backwards();
    void ChanB_Off();

  private:
    int stepsPerRev;
    int Direction_A;
    int PWM_A;
    int Brake_A;
    int Direction_B;
    int PWM_B;
    int Brake_B;
    int state;
};
```

// This is a demo cpp library class that is setup for and WILL work with the Arduino motor shield

CPP body for the motor shield class

```
#include "StepperMS.h"

// construction
StepperMS::StepperMS(int stepsPerRev, int Direction_A, int PWM_A, int Brake_A, int Direction_B, int PWM_B, int Brake_B)
{
    this->stepsPerRev = stepsPerRev;

    this->Direction_A = Direction_A;
    this->PWM_A = PWM_A;
    this->Brake_A = Brake_A;

    this->Direction_B = Direction_B;
    this->PWM_B = PWM_B;
    this->Brake_B = Brake_B;
}
```

Constructor
Used to set parameters

// This is a demo cpp library class that is setup for and WILL work with the Arduino motor shield

```
// initialize Arduino control
void StepperMS::Init()
{
    //set motor direction pins
    pinMode(Direction_A, OUTPUT); //CH_A HIGH = forwards
    pinMode(Direction_B, OUTPUT); //CH_B HIGH = forwards

    // set motor brake pins
    pinMode(Brake_A, OUTPUT); // CH_A brake
    pinMode(Brake_B, OUTPUT); //CH_B brake

    // starting state
    state=0;
}
```

Initialization member function
Used to reset operating variables and the state each time it is called

CPP body for the motor shield class

```
// full step stepper control
void StepperMS::fullStepSinglePhaseCCW()
{
  // decode the state
  switch (state)
  {
    case 0:
      ChanB_Off();
      ChanA_Forwards();
      // update state
      state = 1;
      break;
    case 1:
      ChanA_Off();
      ChanB_Backwards();
      // update state
      state = 2;
      break;
    case 2:
      ChanB_Off();
      ChanA_Backwards();
      // update state
      state = 3;
      break;
    case 3:
      ChanA_Off();
      ChanB_Forwards();
      // update state
      state = 0;
      break;
  }
}
```

// This is a demo cpp library class that is setup for and WILL work with the Arduino motor shield

State machine member function for full step single phase in CCW direction
Each time this function is called it:

- Decodes the state
- Uses current state
- Updates state

CPP body for the motor shield class

```
// switch on channel A
void StepperMS::ChanA_Forwards()
{
    digitalWrite(Brake_A, LOW);          //Enable CHA
    digitalWrite(Direction_A, HIGH);     //Sets direction of CHA
    analogWrite(PWM_A, 255);             //Move CHA
}
// switch on channel A
void StepperMS::ChanA_Backwards()
{
    digitalWrite(Brake_A, LOW);          //Enable CHA
    digitalWrite(Direction_A, LOW);      //Sets direction of CHA
    analogWrite(PWM_A, 255);             //Move CHA
}
// switch off channel A
void StepperMS::ChanA_Off()
{
    digitalWrite(Brake_A, HIGH);         //Disable CH A
}
// switch on channel B
void StepperMS::ChanB_Forwards()
{
    digitalWrite(Brake_B, LOW);          //Enable CHB
    digitalWrite(Direction_B, HIGH);     //Sets direction of CHB
    analogWrite(PWM_B, 255);             //Move CHB
}
// switch on channel B
void StepperMS::ChanB_Backwards()

    digitalWrite(Brake_B, LOW);          //Enable CHB
    digitalWrite(Direction_B, LOW);      //Sets direction of CHB
    analogWrite(PWM_B, 255);             //Move CHB
}
// switch off channel B
void StepperMS::ChanB_Off()
{
    digitalWrite(Brake_B, HIGH);         //Disable CH B
}
```

// This is a demo cpp library class that is setup for and WILL work with the Arduino motor shield

Private member functions to switch on appropriate stepper coils and set or reset brake

ROCO222: Intro to sensors and actuators

Lecture 5

Arduino stepper library

Arduino stepper class

This function creates a new instance of the Stepper class that represents a particular stepper motor attached to your Arduino board. Use it at the top of your sketch, above setup() and loop().

The number of parameters depends on how you've wired your motor - either using two or four pins of the Arduino board.

Parameters

Stepper(steps, pin1, pin2, pin3, pin4)

steps: the number of steps in one revolution of your motor. If your motor gives the number of degrees per step, divide that number into 360 to get the number of steps (e.g. 360 / 3.6 gives 100 steps). (*int*)
pin1, pin2: two pins that are attached to the motor (*int*)
pin3, pin4: *optional* the last two pins attached to the motor, if it's connected to four pins (*int*)

Returns
A new instance of the Stepper motor class.

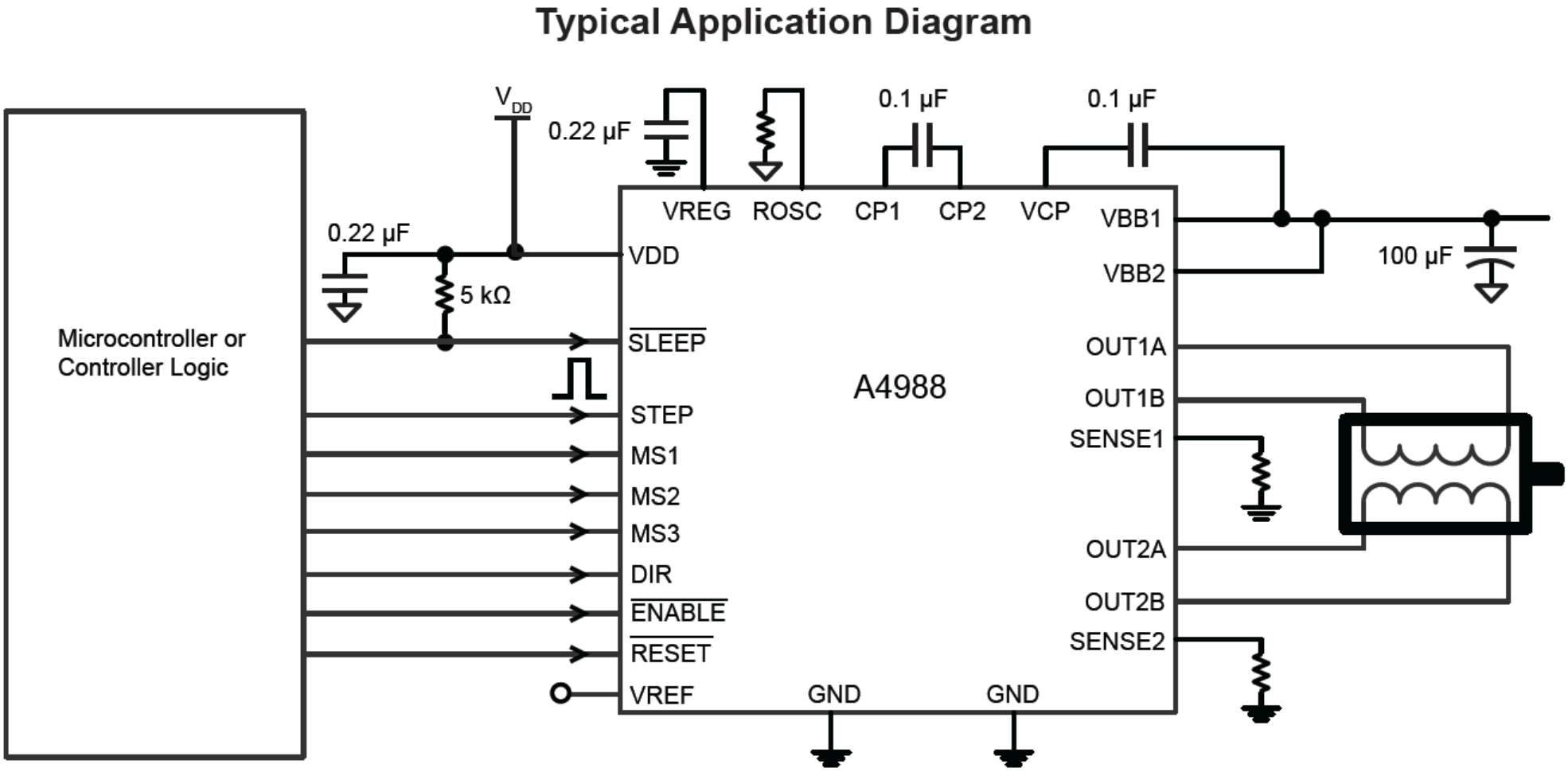
```
/* MotorKnob: A stepper motor follows the turns of a potentiometer on analog input 0  */
// This will not work with the motor shield

#include <Stepper.h>

#define STEPS 100  // change this to the number of steps on your motor

// create an instance of the stepper class, specifying number of steps of motor and pins
// These pins will not work with the motor shield
// it is currently setup to use this pins to interface to a LM298 based controller
// connected to these pins
Stepper stepper(STEPS, 8, 9, 10, 11);

int previous = 0; // the previous reading from the analog input

void setup()
{
    stepper.setSpeed(30); // set the speed of the motor to 30 RPMs
}

void loop()
{
  int val = analogRead(0);    // get the sensor value
```

ROCO222: Intro to sensors and actuators

Lecture 5

Arduino A4988 stepper motor control

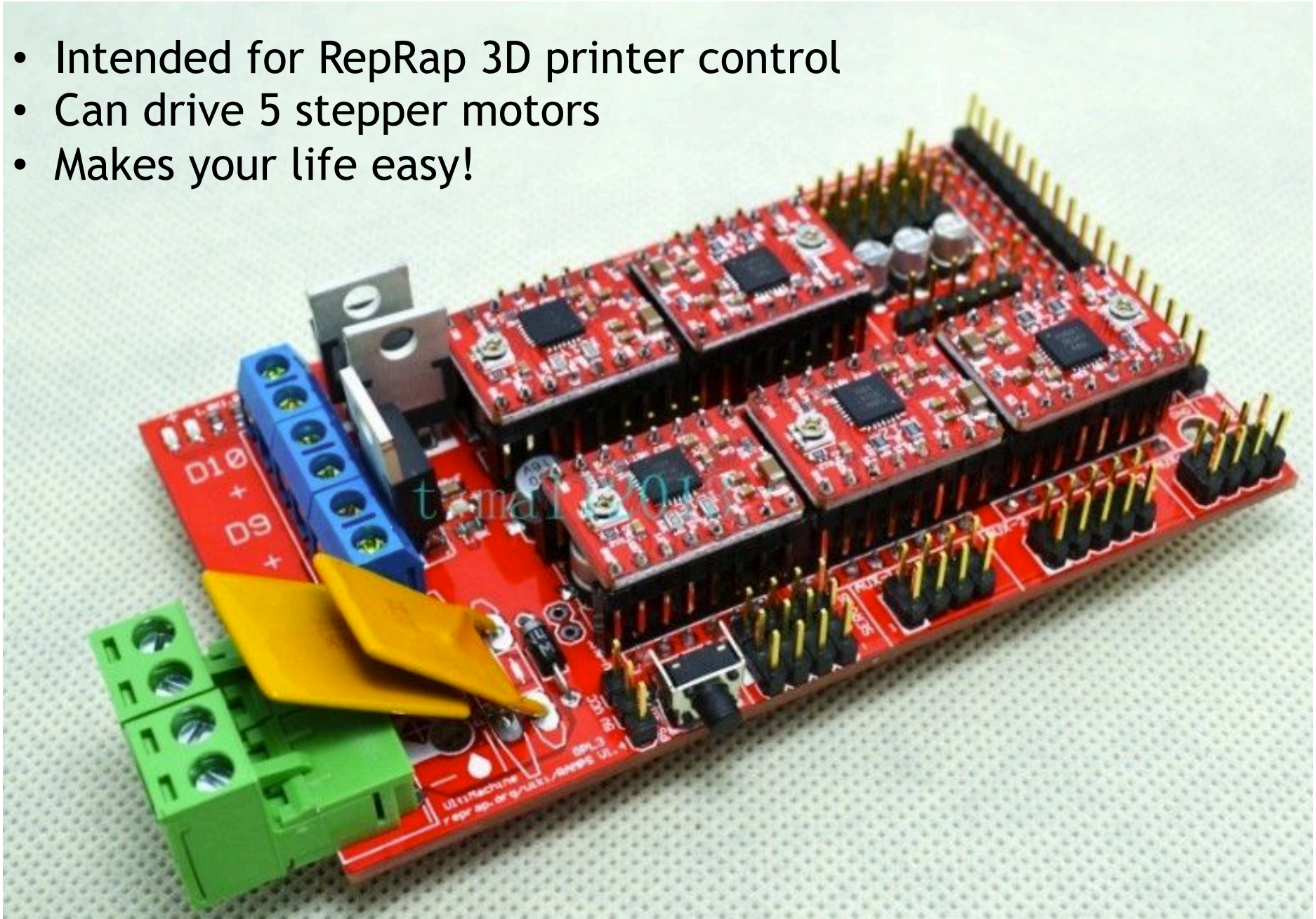
A4988 Stepper Motor Controller



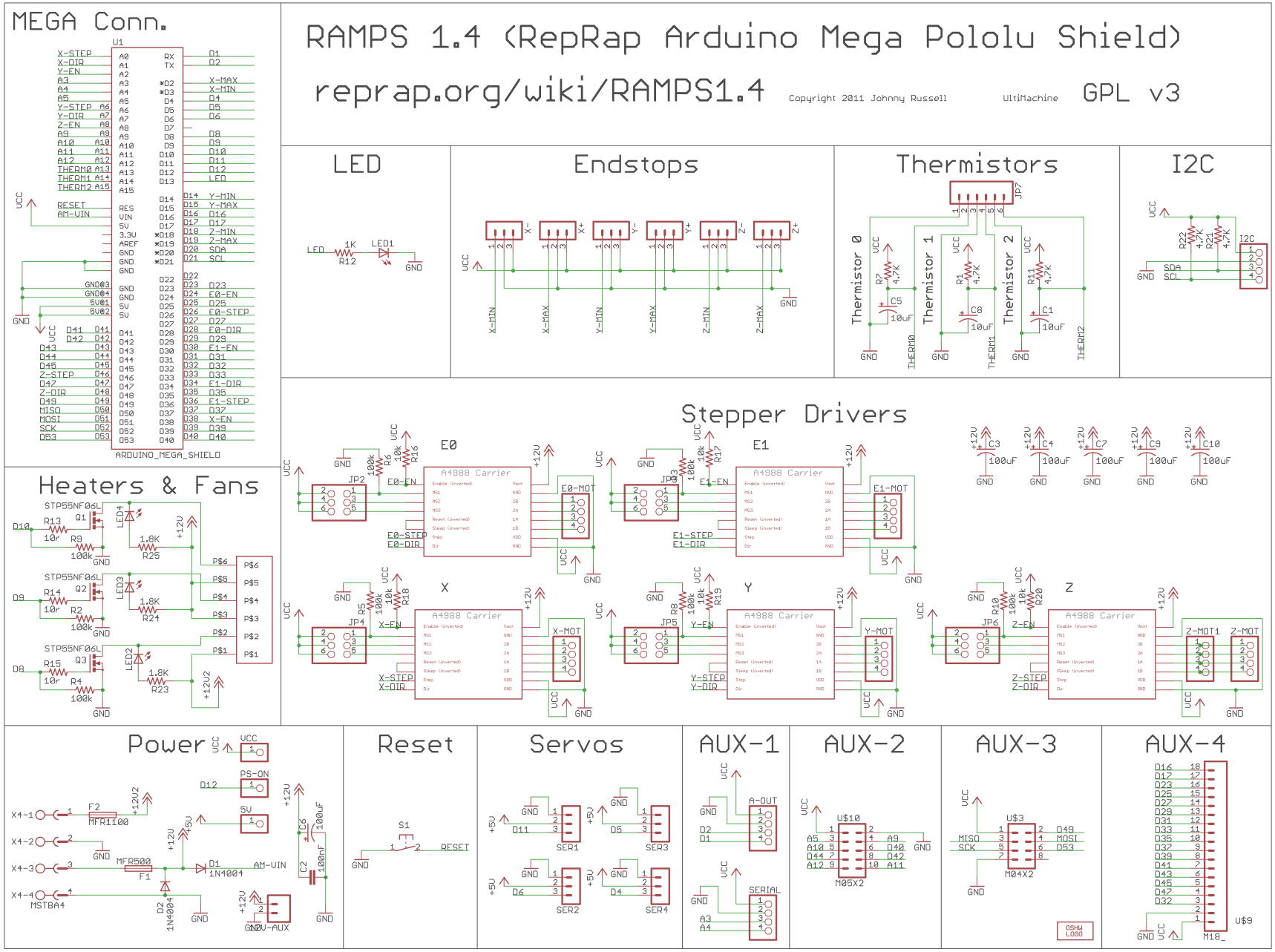
- Essentially the same as the DRV8825 Stepper Motor Controller
- Intelligent current control
- Simple step and direction control interface
- Need to generate control pulse and direction signals

RAMPS 1.4 control board

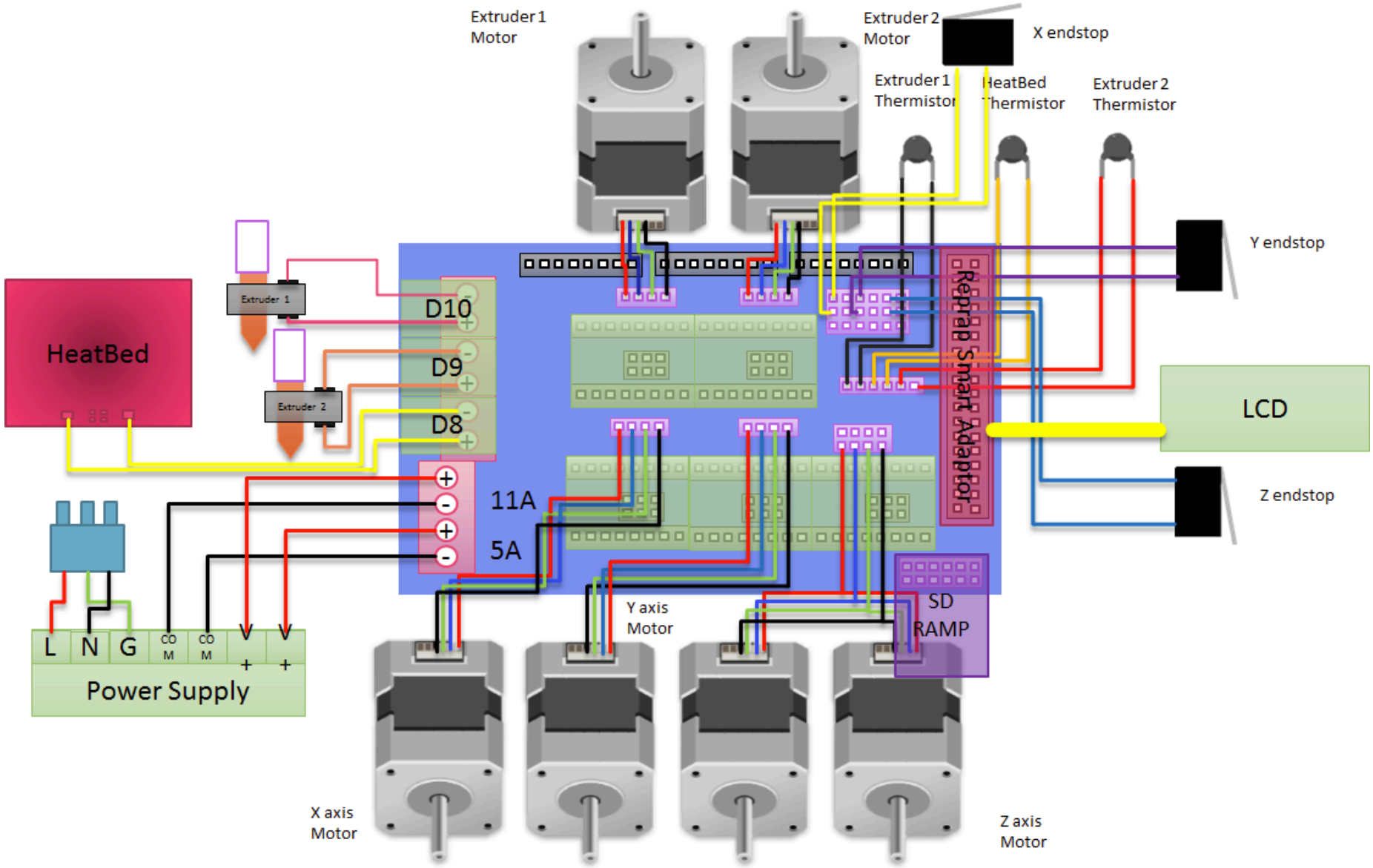
- Intended for RepRap 3D printer control
- Can drive 5 stepper motors
- Makes your life easy!



RAMPS 1.4


MEGA Conn.
RAMPS 1.4 (RepRap Arduino Mega Pololu Shield)
reprap.org/wiki/RAMPS1.4
Copyright 2011 Johnny Russell
UltiMachine
GPL v3
LED
Endstops
Thermistors
I2C
Heaters & Fans
Stepper Drivers
Power
Reset
Servos
AUX-1
AUX-2
AUX-3
AUX-4
ARDUINO_MEGA_SHIELD
A4988 Carrier
E0
E1
X
Y
Z
Thermistor 0
Thermistor 1
Thermistor 2

RAMPS 1.4 control board



OOD Arduino code using A4988

- Simple OOD class to drive a stepper back and forth on X-axis
- Only critical cpp code shown here

```
#include "RAMPS.h"

RAMPS::RAMPS(int loopDelay, int driveSteps, int waitDelay)
{
    this->loopDelay = loopDelay;
    this->driveSteps = driveSteps;
    this->waitDelay = waitDelay;
    this->dirHigh = true;;
}

RAMPS::~RAMPS()
{

}
```

- Constructor to setup parameters

- Init to setup board parameters

```
void RAMPS::SetupXAxis()
{
    // switch on X motor
    pinMode(X_STEP_PIN , OUTPUT);
    pinMode(X_DIR_PIN , OUTPUT);
    pinMode(X_ENABLE_PIN , OUTPUT);
    digitalWrite(X_ENABLE_PIN , LOW);
    dirHigh = true;
    digitalWrite(X_DIR_PIN, HIGH);
    digitalWrite(X_STEP_PIN, LOW);
}
```

- Board pin definitions

```
// For RAMPS 1.4

#define X_STEP_PIN         54
#define X_DIR_PIN          55
#define X_ENABLE_PIN       38
#define X_MIN_PIN           3
#define X_MAX_PIN           2
```

OOD Arduino code using A4988

```
void RAMPS::RunXAxis()
{
    int idx;

    // on
    digitalWrite(X_ENABLE_PIN , LOW);

    // Toggle the DIR pin to change direction.
    if (dirHigh)
    {
        dirHigh = false;
        digitalWrite(X_DIR_PIN, LOW);
    }
    else
    {
        dirHigh = true;
        digitalWrite(X_DIR_PIN, HIGH);
    }
    // step one revolution  in one direction:
    for (idx = 0; idx < driveSteps; idx++)
    {
        digitalWrite(X_STEP_PIN , HIGH);
        delayMicroseconds(loopDelay);
        digitalWrite(X_STEP_PIN , LOW);
        delayMicroseconds(loopDelay);
    }

    // let it cool off!
    digitalWrite(X_ENABLE_PIN , HIGH);
    delay(waitDelay);
}
```

- Function to drive stepper in specified direction for ‘driverSteps’ steps
- Set direction
- Write pulses
- Turn off for a bit

OOD Arduino code using A4988

```
1  // move stepper with simple pulse generation
2
3  // For RAMPS 1.4
4  #include "RAMPS.h"
5
6  // pulse delay
7  // max for 0.9 deg steppers
8  const int loopDelay = 900; // max for 0.9 deg steppers
9  const int driveSteps = 290;
10 const int waitDelay = 50;
11
12 // build RAMPS controller
13 RAMPS oramps = RAMPS(loopDelay, driveSteps, waitDelay);
14
15
16 void setup() {
17
18   // switch on X motor
19   oramps.SetupXAxis();
20
21   // set the speed at maxSpeed rpm:
22   // initialize the serial port:
23   Serial.begin(9600);
24 }
25
26
27 void loop() {
28   oramps.RunXAxis();
29 }
```

- Include the class
- Specify stepper parameters
- Build the stepper object
- Initialize object
- Run the object