Microprocessor Robotics Workshop: Attiny Canbot Tutorial

Parts

Parts List

1x Atmel Attiny85
2x HXT900 from Hobbyking or any same sized micro servo
1x 3.7V LiPo battery (e.g., http://www.all-battery.com/3.7V600mAh25CLIPORechargeableBattery-34062.aspx)
1x HC-SR04 Ultrasonic range sensor
1x IR and Remote (https://www.amazon.com/gp/product/B01EE4VXS0/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1)
1x 9 x 4mm switch
Perf board
3D printed parts: http://www.thingiverse.com/thing:965449

For programming:

Computer
Arduino Uno and USB cable
Breadboard and jumper cables
10 µF capacitor
LED

Tools:

Soldering iron
Hot glue gun

Links

Instructables Canbot: http://www.instructables.com/id/Attiny-Canbot/

Coretech Robotics Canbot: http://coretechrobotics.blogspot.com/2015/12/attiny-canbot.html
Robotshop Canbot: https://www.robotshop.com/letsmakerobots/attiny-can-robot
Robot parts files: https://www.thingiverse.com/thing:965449
Programming an ATtiny with an Arduino: http://highlowtech.org/?p=1695 and http://highlowtech.org/?p=1706
ArduinoISP tutorial: https://www.arduino.cc/en/Tutorial/ArduinoISP
About servos: https://www.arduino.cc/en/Reference/ServoWrite

Steps

3D print the robot parts

https://www.thingiverse.com/thing:965449 -- Top, bottom, and two wheels



Programming the ATtiny85

Gather the following materials:

- Arduino Uno
- ATtiny85
- 10 uF capacitor
- breadboard
- jumper wires
- LED

In this workshop we will wire up an Arduino and use it as an ISP (programmer) for our ATtiny85. There are many other options for programming your chip, including for example the Tiny AVR Programmer from SparkFun or the AVRISP mkII, but we figured you might enjoy having an Arduino to use for a variety of future projects. Plus, the Arduino can act as a power supply for your board while you're prototyping.

In order to program with the Arduino, we need to do three things:

1. Program the Arduino to program other chips, by uploading the ArduinoISP sketch
2. Telling the Arduino IDE software about the ATtiny85 chip by installing a board definition, so that it knows how to communicate with it
3. Connect the Arduino to the ATtiny85 chip and upload our robot program

Details are outlined below.

Download and Install the Arduino Software

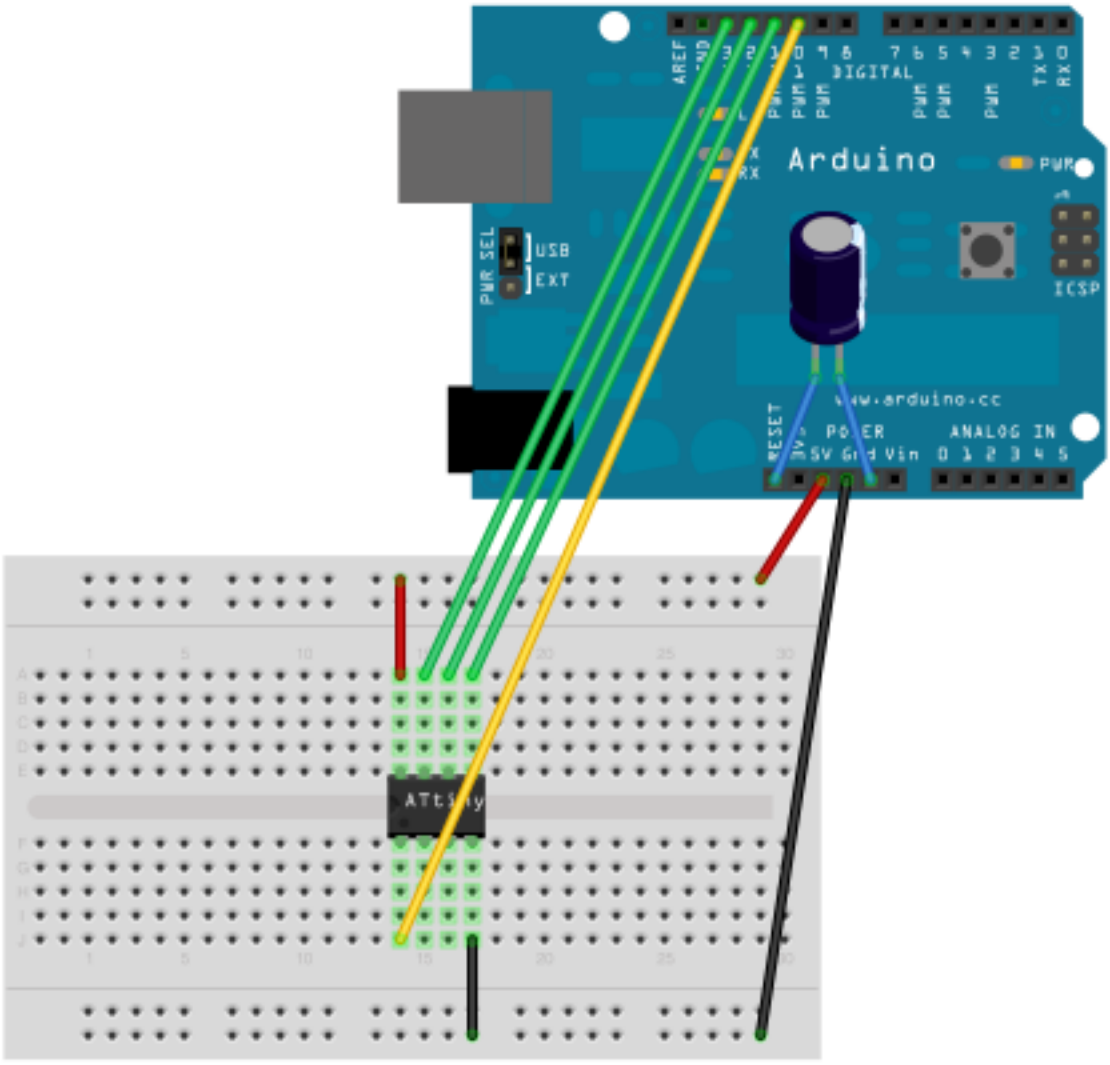
Download and install the Arduino IDE. You should have version 1.8.5. (note to Linux users: download the software from the site, NOT from a package manager.)

Turn the Arduino board into a programmer

- Set up your Arduino
    - Plug your Arduino into the computer using the USB cable
    - Run the Arduino development environment.
    - Under Tools > Board select Arduino Uno
    - Under Tools > Serial Port select the port where the Arduino is connected
    - Make sure things are working by uploading an example sketch
        - "File > Examples > 01. Basics > Blink" is a good one -- after uploading the amber LED on the board should blink on and off
- Turn your Arduino into a programmer
    - Open the ArduinoISP sketch from the Examples menu.
    - Go to Sketch > Upload to upload the ArduinoISP sketch
    - Verify that the sketch uploaded successfully (that is, there are no errors)
- Add the ATtiny85 board definition to the Arduino IDE
    - Open File > Preferences and look for the "Additional Board Manager URLs" at the bottom
    - Paste the following URL into that field:
        - https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json
    - Click OK to save your preferences
    - Go to Tools > Board > Boards Manager
    - Look for **atflash** in the list and click on it
    - Click the Install button to install the latest version
    - Close the Boards Manager. You should now see an entry for the ATtiny in the Tools > Board menu

Connect the Arduino board and the ATtiny

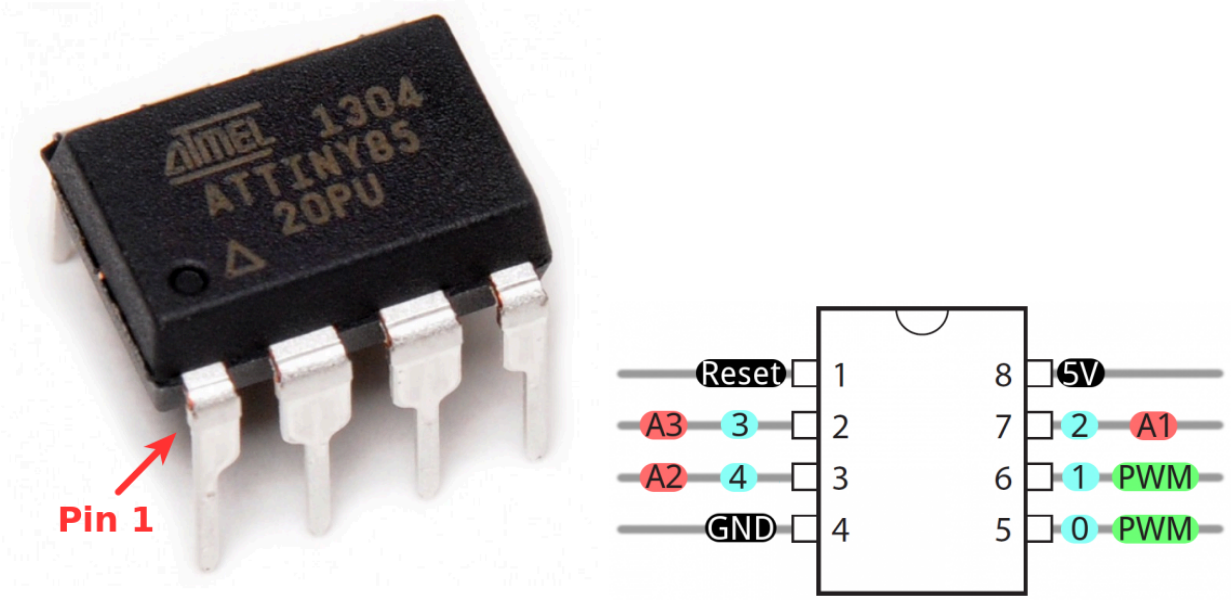
Connect the Arduino board to the ATtiny as shown in the following diagram. Use the dot in the corner of the ATtiny to orient it properly (the dot is Pin 1). Connect a 10 uF capacitor between reset and ground on the Arduino board as shown in the diagram (the stripe on the capacitor that's marked with a negative sign (-) should go to ground). The capacitor prevents the Arduino board from resetting (which starts the bootloader), thus ensuring that the Arduino IDE talks to the ArduinoISP (not the bootloader) during the upload of sketches.



NOTE ABOUT PINS: Looking at the diagram below, you can see that there are two different pin naming schemes used. The first one is just the standard IC naming, 1 through 8. The second naming is Resent, Gnd, 5V, and pin 0, 1, 2, 3, 4. In this document, the pin numbers refer to pins 1-8.

Pin connections:

- **Capacitor to Arduino Reset and GND (-)**
- Arduino GND and +5V to breadboard
- ATtiny Pin 1 to Arduino Pin 10 (RESET)
- ATtiny Pin 4 to GND
- ATtiny Pin 8 to +5V
- ATtiny Pin 7 to Arduino Pin 13 (SCK)
- ATtiny Pin 6 to Arduino Pin 12 (MISO)
- ATtiny Pin 5 to Arduino Pin 11 (MOSI)



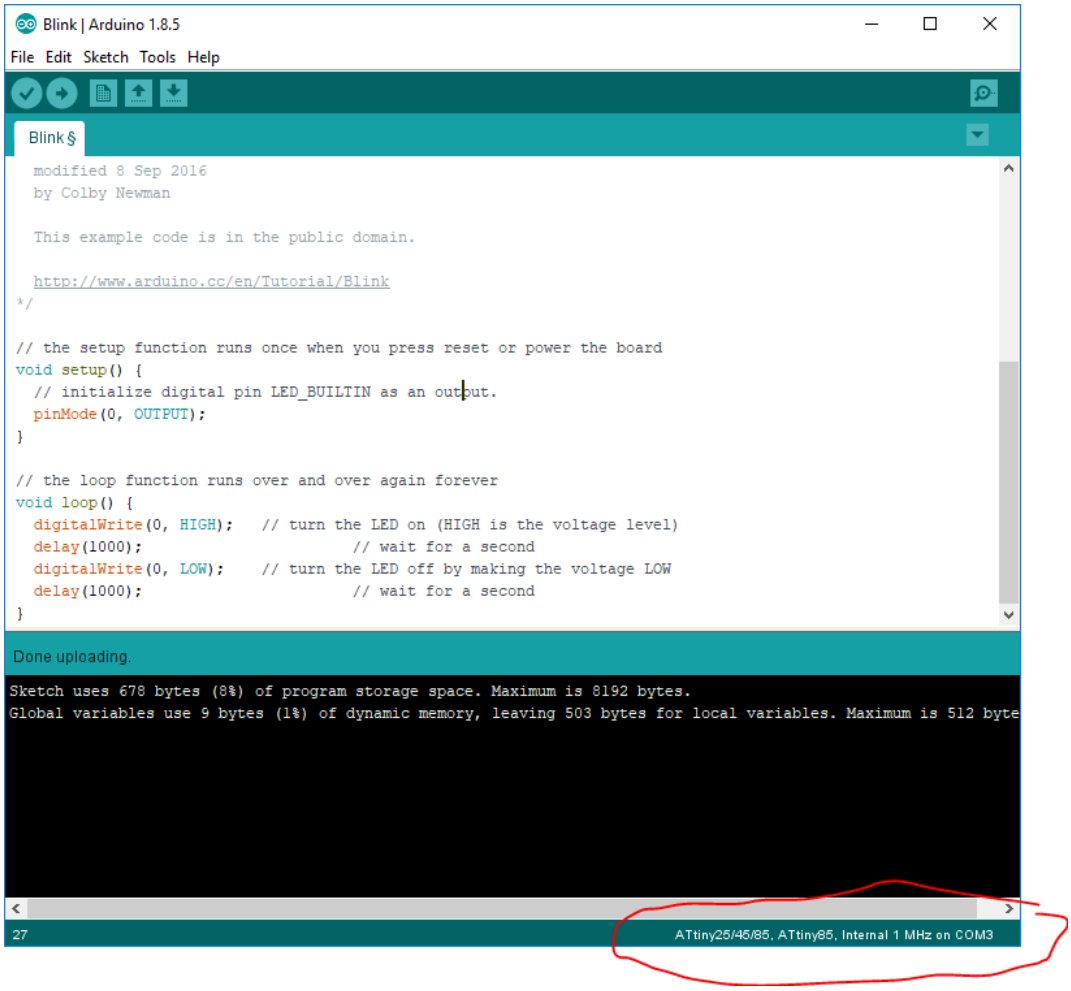
ATtiny85. Pinout on right shows top view

Program the ATtiny using the Arduino

At this point, the Arduino IDE is set up and the board is wired to program our chip. Before we upload our main CanBot program, we'll test that things are working by blinking an LED.

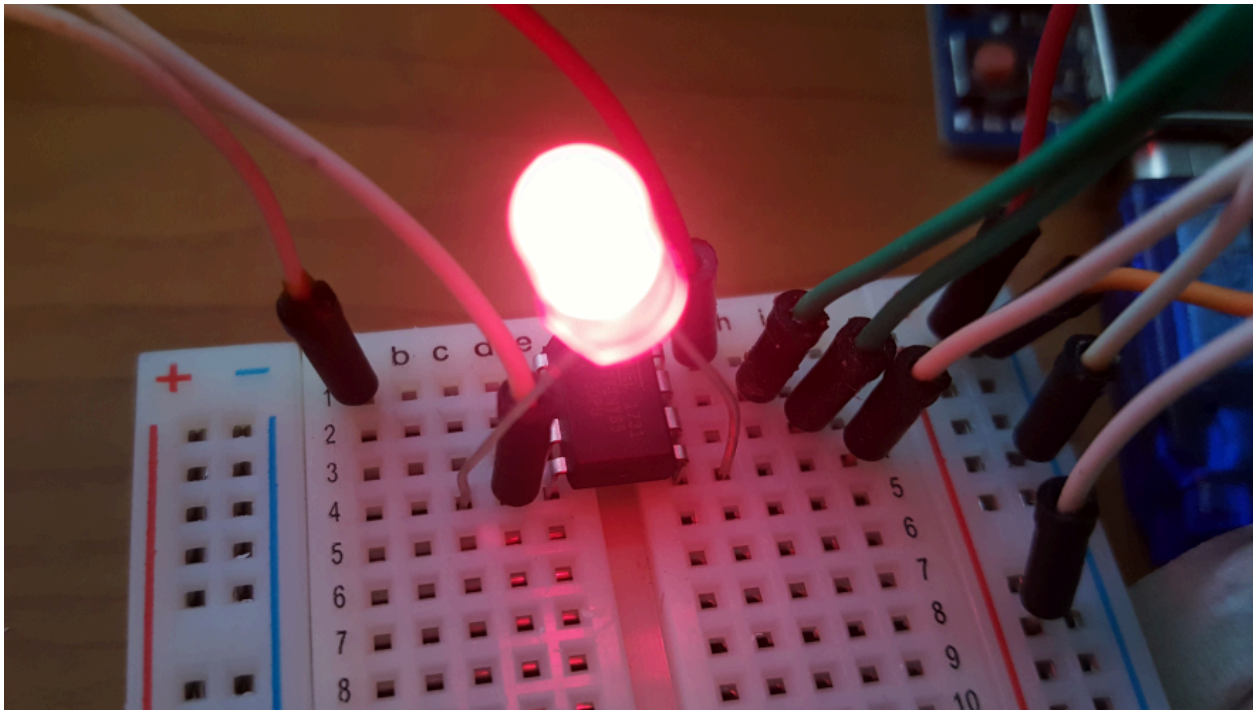
Steps:

- Open the Blink sketch from the Examples menu (File > Examples > 01. Basics > Blink)
- Replace `LED_BUILTIN` with 0 (this changes it from the Arduino built-in LED at pin 13 to pin 0 on the attiny)
- Set options:
    - Select "ATtiny 25/45/85" from the Tools > Board menu
    - Choose "ATtiny 85" from the Tools > Processor menu
    - Choose "Internal 1 MHz" from the Tools > Clock menu
    - Select "Arduino as ISP" from the Tools > Programmer menu
- Upload the sketch



You should see “Done uploading.” in the Arduino software and no error messages. If you then connect an LED between pin 5 and ground, you should see it blink on and off for **one second each**. (Note that you may need to disconnect the LED before uploading a new program.)

If you do see error messages, go to File > Preferences and check the “upload” checkbox next to “Show verbose output during” and try again.



A successful test!

At this point, your Arduino is ready to upload code!

Robot Code

You can download three different projects from the Classes page on the AMS website: http://www.anchoragemakerspace.com/classes/ -- look under Microprocessor Robotics Workshop for Canbottiny_ServoCalibrate, Canbottiny_auto, and Canbottiny_remote. Download all three of these and extract them somewhere on your computer.

- Canbottiny_servoCalibrate -- this has a simple servo calibration script (see section below)
- Canbottiny_auto -- this script runs the robot in autonomous mode, with some simple obstacle avoidance
- Canbottiny_remote -- this more complex script reads data from the IR sensor and allows you to remote control the robot

You should start with the simpler autonomous script, and if you get that working move on to the remote script.

Uploading code to the ATtiny85

Before uploading, make sure your board is in the same configuration as described in the programming section above. It might be helpful to have two breadboard setups -- one for programming and one for the actual robot circuit.

- Load the code into the Arduino IDE
    - Go to File > Open and choose Canbottiny_XXX.ino inside the folder you downloaded
    - It should open 3 files in the IDE
- Upload to the attiny85
    - Make sure everything is still wired up like in the step above, and that you still have the ATtiny85 as your board and Arduino as ISP as your programmer
    - Remove the LED, if it's still connected
    - Hit Upload in the Arduino IDE
    - It should upload with no errors

You can then move the chip from your programming breadboard to your robot breadboard to test it out.

Calibrate Servos

We are using continuous rotation servos for this project, and a small library called Servo8Bit. This is different than the standard Arduino Servo library since it's for running on the ATtiny85, however the basic methods are the same:

- Use servo.write(angle) to control the servos
    - angle=90 -- no motion
    - angle=0 -- full clockwise
    - angle=180 -- full counter-clockwise

The servos have a small potentiometer that you can use to calibrate them. Out of the box, you could call servo.write(90) and depending on the servo there could be no motion, some motion, or a lot of motion! So we need to use a small calibration script and the potentiometer to make sure everything is set properly.

1. Program the tiny with the Canbottiny_servoCalibrate code below
2. Wire up the servos on the breadboard, using the electronics diagrams below
3. Plug in the power to start the code running

4. If the servos move, turn the pot screw on the back of the servo until they don't move anymore

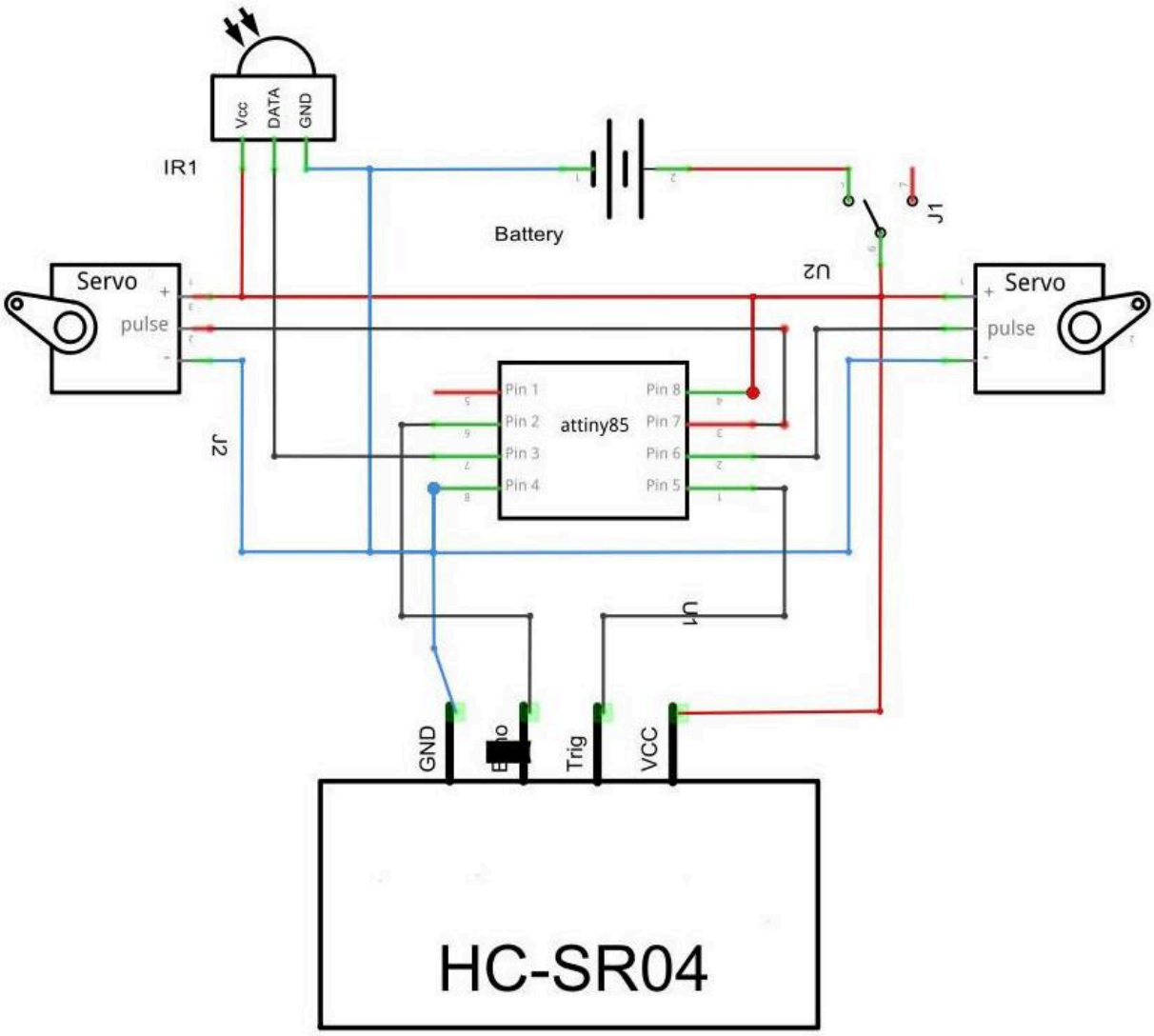
---

Electronics

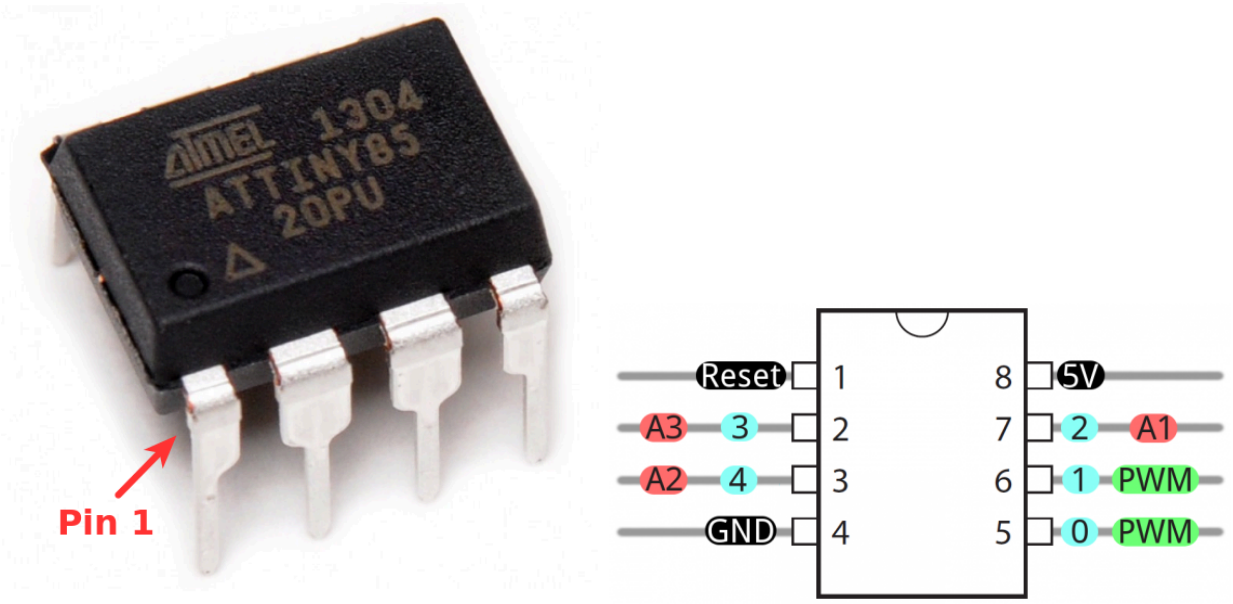
Steps:

1. Gather all components
2. Build circuit using breadboard
    a. Disconnect your Arduino from the computer
    b. Create circuit below using a breadboard and jumper cables
    c. Connect GND on your breadboard to GND on your Arduino (leave the +5V off for now)
    d. Plug the Arduino into your computer to give it power
3. Test the circuit
    a. Make sure your chip is programmed with the Canbottiny code
    b. Connect +5V on your breadboard to +5V on the Arduino
    c. Move your hand back and forth in front of the distance sensor. The motors should respond accordingly
4. Solder components to perf board
5. Check for shorts!
6. Plug in the battery and test again

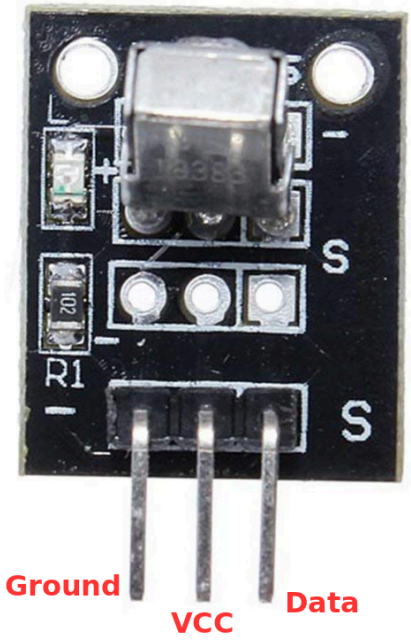
If everything is working well, you're ready for assembly!



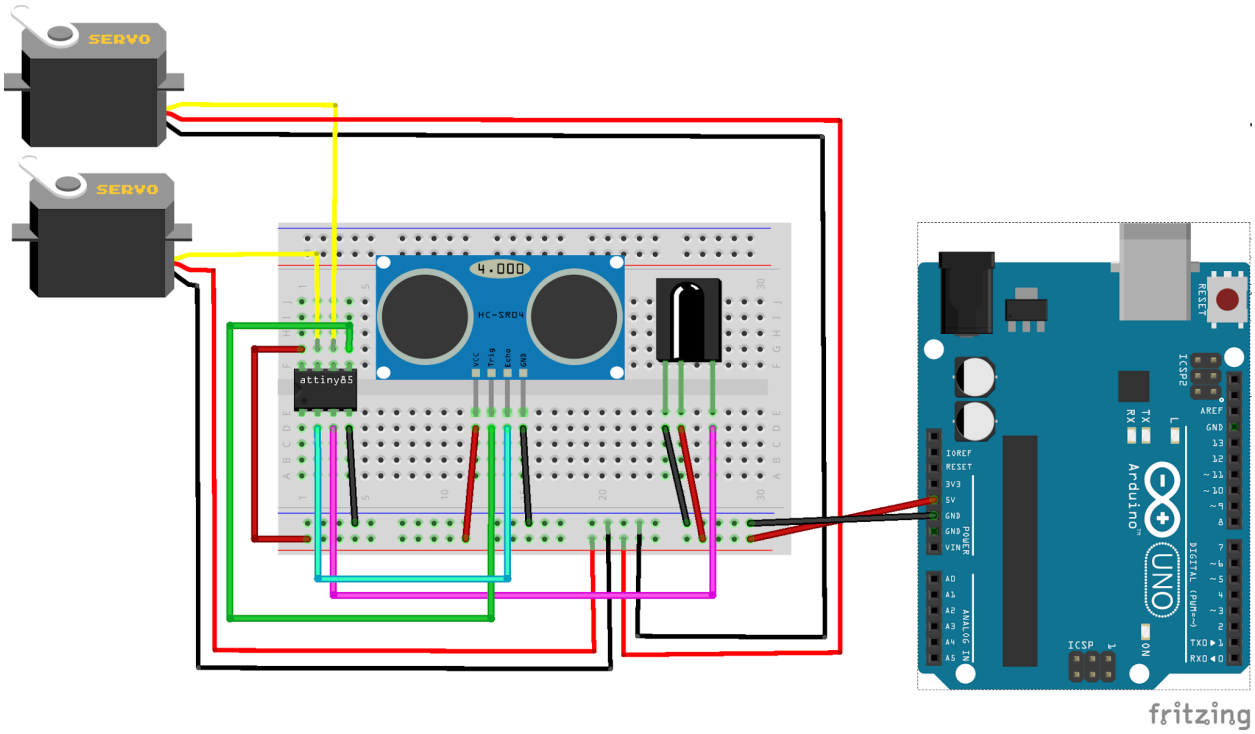
Main diagram



ATtiny85. Pinout on right shows top view



IR Sensor. Note that pins are ordered differently than in main diagram above



Breadboard diagram

Assembly

