Pololu 3pi+ 32U4 User’s Guide

1. Overview . . . 3
    1.1. Configurations and included components . . . 6
    1.2. What you will need . . . 13
    1.3. Supported operating systems . . . 14
2. Contacting Pololu . . . 15
3. Assembling the 3pi+ 32U4 kit . . . 16
4. Using the preloaded demo program . . . 37
5. The 3pi+ 32U4 in detail . . . 40
    5.1. Microcontroller . . . 41
    5.2. User interface . . . 42
    5.3. Motors . . . 44
    5.4. Quadrature encoders . . . 46
    5.5. Line and bump sensors . . . 48
    5.6. Inertial sensors . . . 50
    5.7. Power . . . 51
    5.8. Expansion headers . . . 55
    5.9. Pin assignments . . . 57
    5.10. Adding electronics . . . 61
    5.11. AVR timers . . . 63
    5.12. Schematics and dimensions . . . 63
6. Programming the 3pi+ 32U4 . . . 65
    6.1. Installing Windows drivers . . . 65
    6.2. Programming using the Arduino IDE . . . 67
    6.3. Programming using avr-gcc and AVRDUDE . . . 72
7. Pololu3piPlus32U4 Arduino library . . . 75
8. The 3pi+ 32U4 USB interface . . . 76
9. The A-Star 32U4 Bootloader . . . 78
10. Reviving an unresponsive 3pi+ 32U4 . . . 81
    10.1. Reviving using the Arduino IDE . . . 81
    10.2. Reviving using AVRDUDE . . . 84
11. Related resources . . . 85

1. Overview

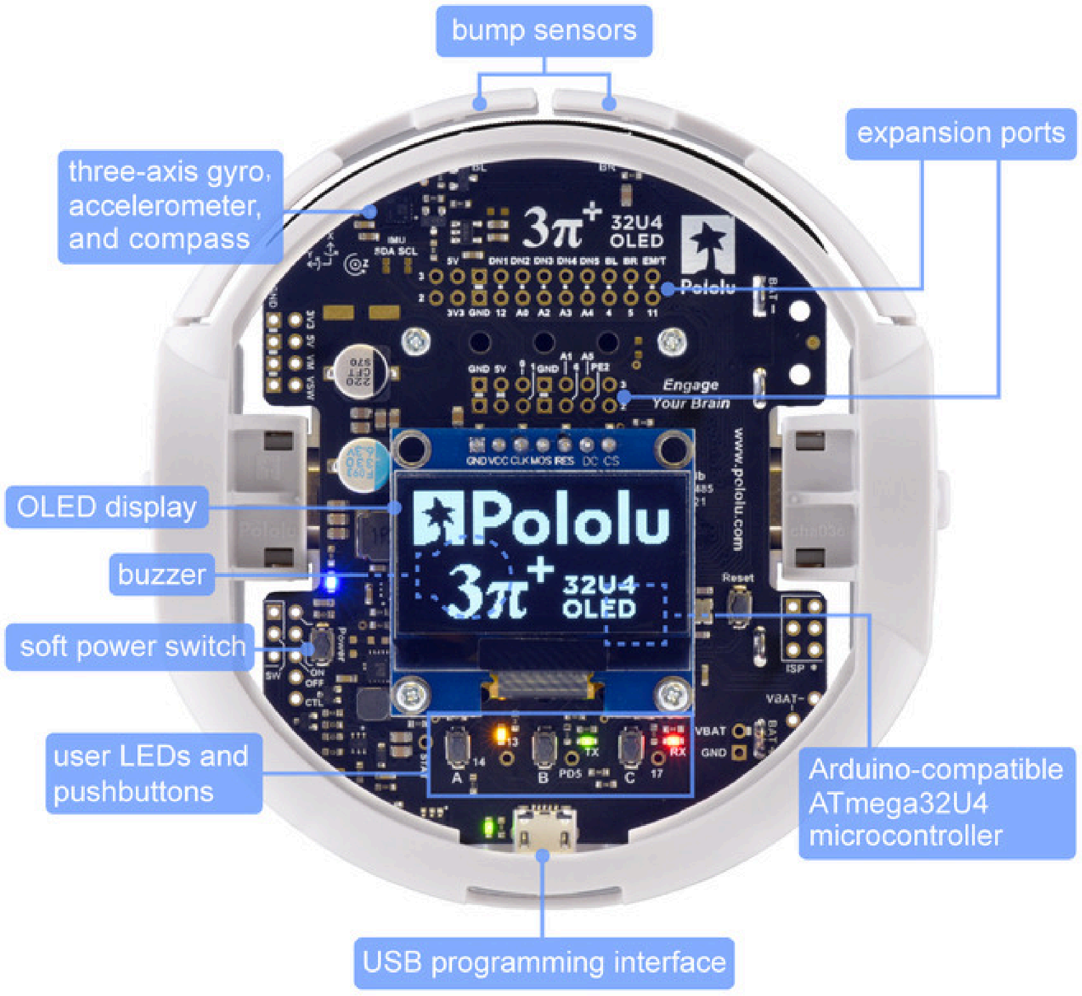
The 3pi+ 32U4 is a versatile, high-performance, user-programmable robot that measures just 9.7 cm (3.8″) in diameter. At its heart is an ATmega32U4 AVR microcontroller from Microchip (formerly Atmel), and like our **A-Star 32U4 programmable controllers [https://www.pololu.com/category/149/a-star-programmable-controllers]**, the 3pi+ 32U4 features a USB interface and ships preloaded with an Arduino-compatible bootloader, so all you need to program it is a **USB A to Micro-B cable [https://www.pololu.com/product/2072]** (not included). A software add-on is available that makes it easy to program the robot from the Arduino environment, and we have Arduino libraries and example sketches to help get you started. For advanced users who want to customize or enhance their robots with additional peripherals, the robot’s power rails, power system controls, and microcontroller’s I/O lines can be accessed via several 0.1″-pitch expansion ports.

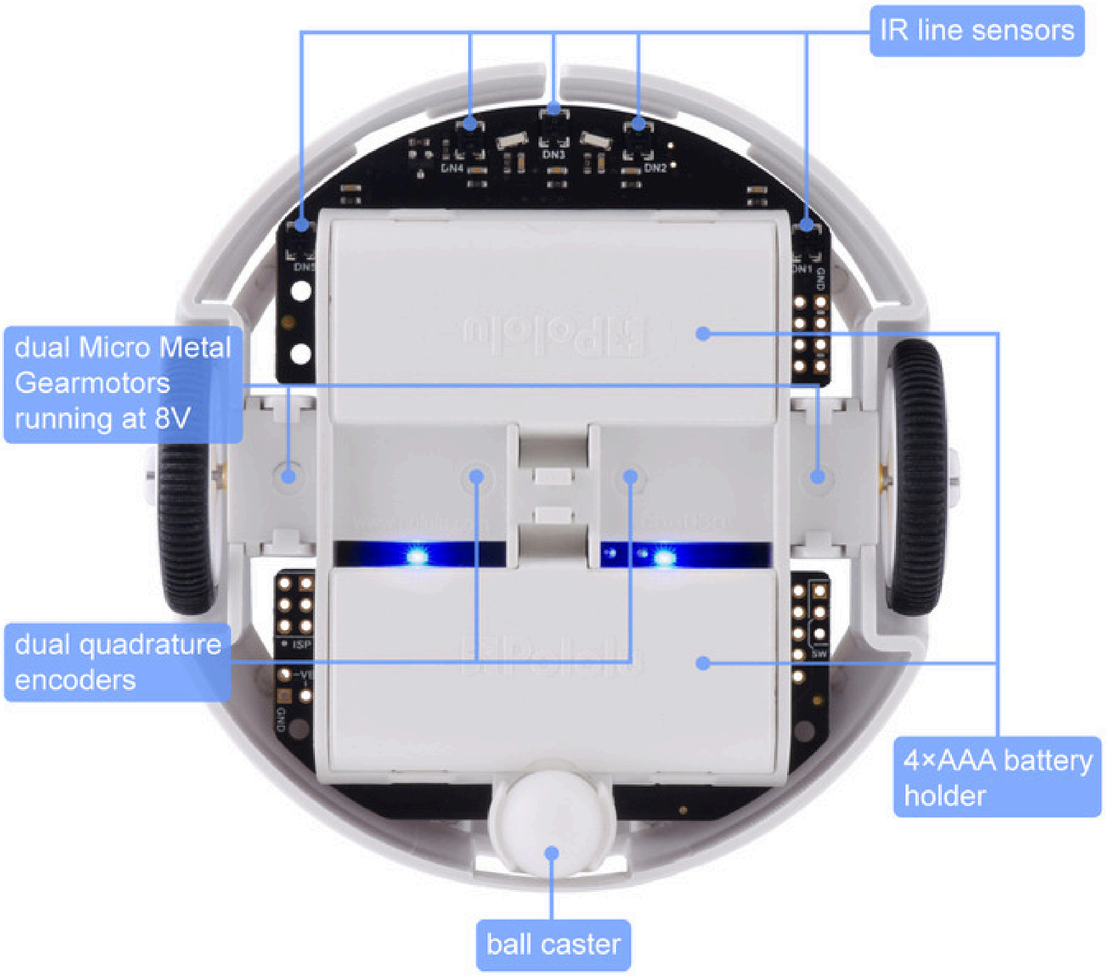
3pi+ 32U4 OLED Robot, front view.

3pi+ 32U4 OLED Robot, side view.

3pi+ 32U4 OLED Robot, rear view.

The 3pi+ (or 3π+) features two H-bridge motor drivers and a variety of integrated sensors, including a pair of quadrature encoders for closed-loop motor control, a complete inertial measurement unit (3-axis accelerometer, gyro, and magnetometer), five downward-facing reflectance sensors for line-following or edge-detection, and left and right bump sensors along the front face of the robot. Three on-board pushbuttons offer a convenient interface for user input, and a 128×64 graphical OLED display (LCD on original version), buzzer, and indicator LEDs allow the robot to provide feedback.


bump sensors
expansion ports
three-axis gyro, accelerometer, and compass
OLED display
buzzer
soft power switch
user LEDs and pushbuttons
Arduino-compatible ATmega32U4 microcontroller
USB programming interface



The robot also features a unique power system that runs the motors at a constant 8 V independent of the battery charge level, so the speed of the motors does not vary with the battery voltage.

3pi+ 32U4 versions

The original version of the 3pi+ 32U4 includes an **8×2 character LCD [https://www.pololu.com/product/356]**, while the newer **3pi+ 32U4 OLED** incorporates a **graphical OLED display [https://www.pololu.com/product/3760]** instead. The information in this user's guide generally applies to **both versions**, and the name "3pi+ 32U4" covers both the original (LCD) and OLED versions except where specific differences are noted.

Our **Pololu3piPlus32U4 Arduino library [https://www.pololu.com/docs/0J83/7]** generally allows code written for the LCD version to work on the OLED version with minimal changes (and the reverse is also true as long as your code does not make use of the OLED's graphical capabilities).

3pi+ 32U4 Robot, top view.

3pi+ 32U4 OLED Robot, top view.

Comparison with the original 3pi

The 3pi+ 32U4 offers many major improvements over the **original 3pi** **[https://www.pololu.com/product/975]**, including:

- ATmega32U4 microcontroller with Arduino-compatible bootloader can be programmed directly through a USB connection
- Quadrature encoders on both motors for closed-loop position and speed control
- Graphical OLED display *(OLED version only)*
- Full 9-axis IMU (three-axis gyro, accelerometer, and compass)
- Bottom-loading battery holders keep batteries accessible even if additional levels are added
- Full wrap-around bumper to protect electronics from collisions
- Two bump sensors on the front

The 3pi+ 32U4 uses a different microcontroller with different pin mappings compared to the older 3pi, and some of the software library interfaces are different, so programs written for the original 3pi will not work on the 3pi+ 32U4 without modification.

1.1. Configurations and included components

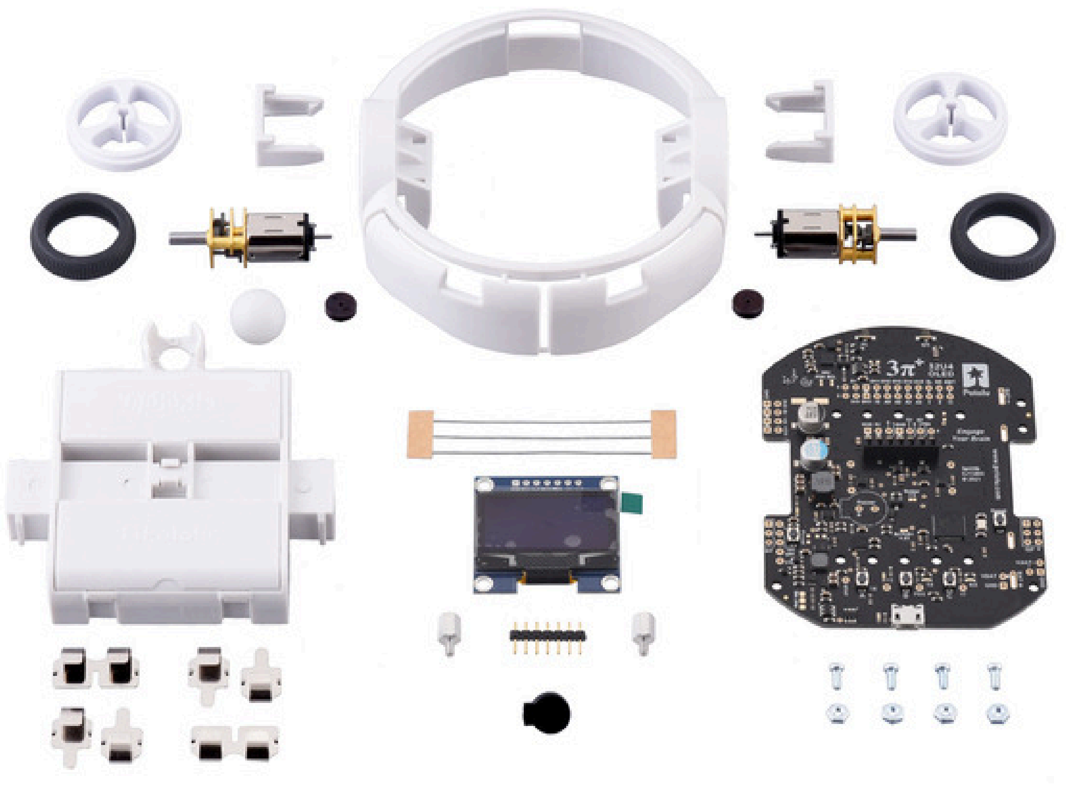
The 3pi+ 32U4 OLED robot is available as a kit or fully assembled with three different motor options:

3pi+ 32U4 OLED Version	Products	Micro Metal Gearmotor	Top Speed	Comments
Standard Edition	**assembled** or **kit**	**30:1 MP 6V**	1.5 m/s	good combination of speed and controllability
Turtle Edition	**assembled** or **kit**	**75:1 LP 6V**	0.4 m/s	longest battery life, easiest to control, good for swarm robots or introductory robotics courses
Hyper Edition	**assembled** or **kit**	**15:1 HPCB 6V**	~4 m/s	**very fast and difficult to control, easy to damage**; only recommended for advanced users

In addition, the **3pi+ chassis [https://www.pololu.com/product/3725]** and **3pi+ 32U4 OLED control board [https://www.pololu.com/product/4974]** are available separately and can be combined with the **motors [https://www.pololu.com/category/141/micro-metal-gearmotors-with-extended-motor-shafts]** of your choice to make a custom kit.

The original (LCD) version of the 3pi+ 32U4 robot is also available in these three editions—Standard (**assembled [https://www.pololu.com/product/3737]** or **kit [https://www.pololu.com/product/3740]**), Turtle (**assembled [https://www.pololu.com/product/3738]** or **kit [https://www.pololu.com/product/3741]**), and Hyper (**assembled [https://www.pololu.com/product/3736]** or **kit [https://www.pololu.com/product/3739]**)—and as a separate **control board [https://www.pololu.com/product/3735]**.

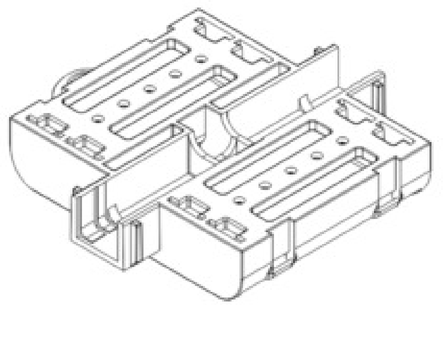
3pi+ 32U4 robot kit contents

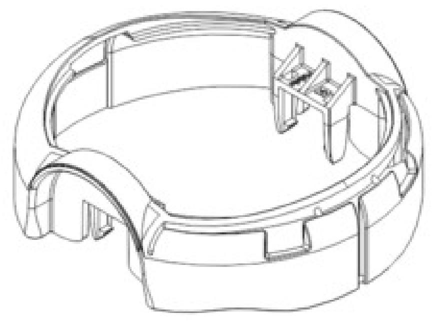


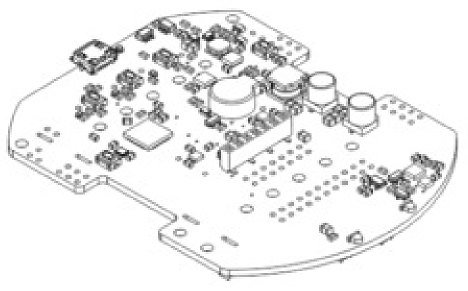
The **kit versions** of the 3pi+ 32U4 OLED robot include the following items:

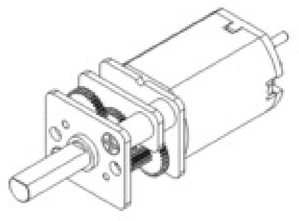
- **3pi+ Chassis Kit [https://www.pololu.com/product/3725]**, which includes:
    - 3pi+ chassis, with integrated 4×AAA battery holder and ball caster
    - 3pi+ bumper skirt
    - two 3pi+ motor clips
    - two 32×7 mm Pololu wheel hubs and silicone tires
    - four single AAA battery contacts
    - two double AAA battery contacts
    - 1/2″-diameter plastic ball
- **3pi+ 32U4 OLED Control Board [https://www.pololu.com/product/4973]**, which includes:
    - control board
    - **buzzer [https://www.pololu.com/product/1484]**
    - jumper wires (for soldering motors to the main board)
    - 1×7 low-profile male header for OLED display *(original version kits include an **2×7***

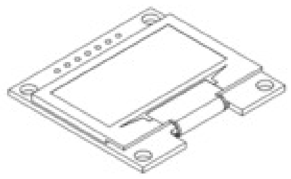
    ***low-profile male header** [https://www.pololu.com/product/2663] for the LCD instead)*
    - two **magnetic encoder discs** [https://www.pololu.com/product/2599] (12 CPR)
    - four **3/16″ #2-56 screws** [https://www.pololu.com/product/2701] and **nuts** [https://www.pololu.com/product/1067]
    - two **1/4″ #2-56 standoffs** [https://www.pololu.com/product/1940] *(OLED version only)*
- two **Micro Metal Gearmotors** [https://www.pololu.com/category/141/micro-metal-gearmotors-with-extended-motor-shafts] (the table above lists the specific motors included with each edition)
- **graphical OLED display** [https://www.pololu.com/product/3760] *(original version kits include an **8×2 character LCD** [https://www.pololu.com/product/356] instead)*
- spare 15:1 gearbox (Hyper Edition only)

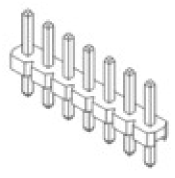

3pi+ chassis

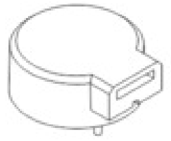

3pi+ bumper skirt

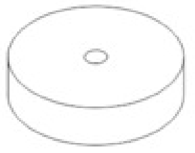

3pi+ 32U4 OLED Control Board

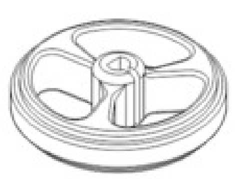

Micro Metal Gearmotor with extended motor shaft ×2

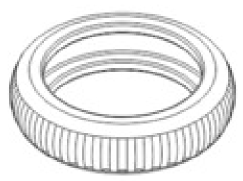

OLED display

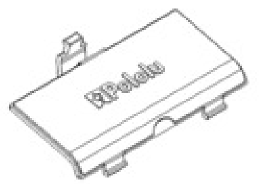

1×7 low-profile male header

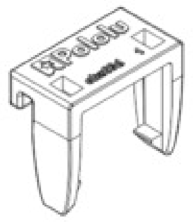

Buzzer

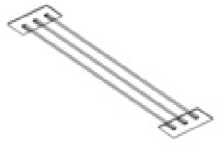

magnetic encoder disc
×2

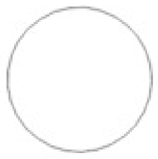

32mm Pololu wheel hub
×2

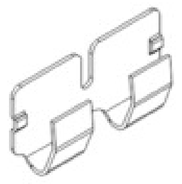

tire for 32mm Pololu wheel
×2

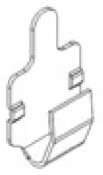

3pi+ battery cover
×2

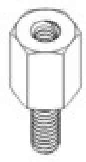

3pi+ motor clip
×2

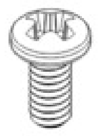

jumper wires

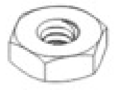

1/2” plastic ball

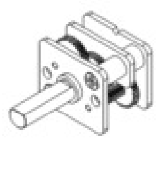

AAA battery contact, double
×2


AAA battery contact, single
×4

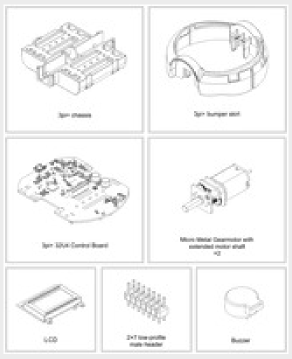

Aluminum standoff,
1/4” length, #2-56 thread
×2

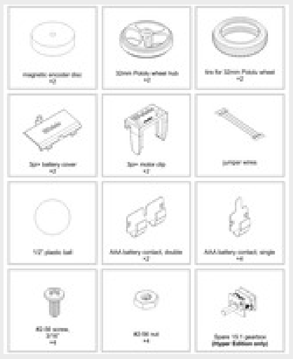

#2-56 screw,
3/16”
×4


#2-56 nut
×4


Spare 15:1 gearbox
(Hyper Edition only)

The diagrams above show the contents of the 3pi+ 32U4 OLED kits. For the contents of the original 3pi+ 32U4 kits, which include an LCD and differ in a few other parts, refer to these diagrams instead.


3pi+ chassis
3pi+ bumper skirt
3pi+ 32U4 Control Board
Micro Metal Gearmotor with extended motor shaft
LCD
2×7 low-profile male header
Buzzer



The robot and chassis kit might include extra parts like jumper wires, screws, and nuts, so do not be concerned if you have some leftover hardware after assembling your 3pi+.

Assembled 3pi+ 32U4 robot

The **assembled versions** of the 3pi+ 32U4 robot are complete, ready-to-program robot platforms built from the same components found in the 3pi+ 32U4 robot kits; no soldering or assembly is required.

1.2. What you will need

These additional items are needed for using the 3pi+ 32U4 robot:

- four AAA batteries—the robot works with both alkaline and NiMH batteries, though we recommend using rechargeable **AAA NiMH cells [https://www.pololu.com/product/1002]**
- **USB A to Micro-B cable [https://www.pololu.com/product/2072]** to connect the robot to your computer for programming and debugging
- small 2 mm slotted screwdriver for adjusting the LCD contrast *(original LCD version only)*

Kit assembly tools

These additional items are needed or helpful for assembling the 3pi+ 32U4 robot kit:

- soldering iron and solder (we recommend one with adjustable temperature control)
- wire cutter
- small #1 Phillips screwdriver

- flush cutter, knife, and/or file for cutting supports on the bumper skirt (optional but recommended)
- Tape or small clamps (for holding parts together when soldering)

Additional optional components

You might also consider getting these for your 3pi+ 32U4 robot:

- **Sensors [https://www.pololu.com/category/7/sensors]**, such as **proximity sensors or range finders [https://www.pololu.com/category/189/proximity-sensors-and-range-finders]**
- **Connectors and jumper wires [https://www.pololu.com/category/19/connectors]**, for connecting additional sensors and components
- Battery charger, if you are using rechargeable batteries; since the 3pi+ just uses ordinary AAA batteries, we recommend basic AAA chargers (into which you stick the individual cells) available at most general electronics stores, though we carry a much fancier **iMAX-B6AC V2 balance charger/discharger [https://www.pololu.com/product/2588]** that can be also used for this

1.3. Supported operating systems

The 3pi+ 32U4 can be programmed using current versions of Microsoft Windows 10, Linux, and macOS. See our **A-Star 32U4 bootloader page on GitHub [https://github.com/pololu/a-star/tree/master/bootloaders/caterina]** for a list of older operating systems that have been tested with the bootloader and are likely to work.

2. Contacting Pololu

We would be delighted to hear from you about any of your projects and about your experience with the 3pi+ 32U4. You can **contact us** **[https://www.pololu.com/contact]** directly or post on our **forum** **[http://forum.pololu.com/]**. Tell us what we did well, what we could improve, what you would like to see in the future, or anything else you would like to say!

3. Assembling the 3pi+ 32U4 kit

This section explains how to assemble the kit version of the 3pi+ 32U4 robot. If you have the pre-assembled version, you can skip to **Section 4**.

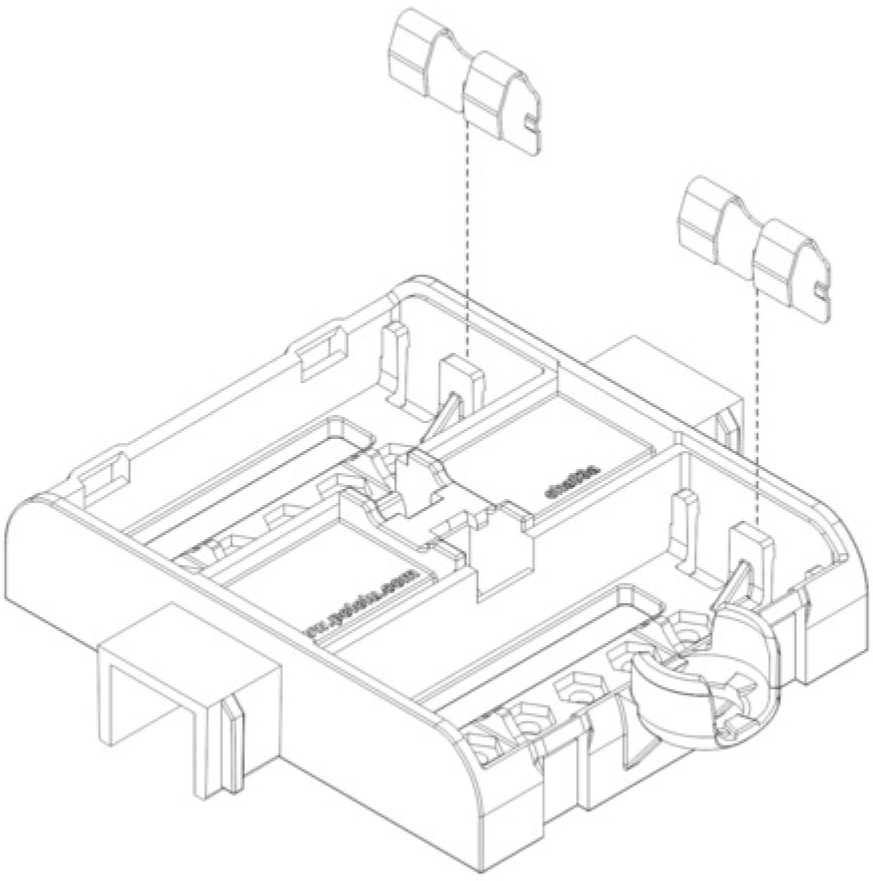
See **Section 1.1** for a diagram to help you identify the contents of the 3pi+ 32U4 robot kit.

Testing the control board before assembly

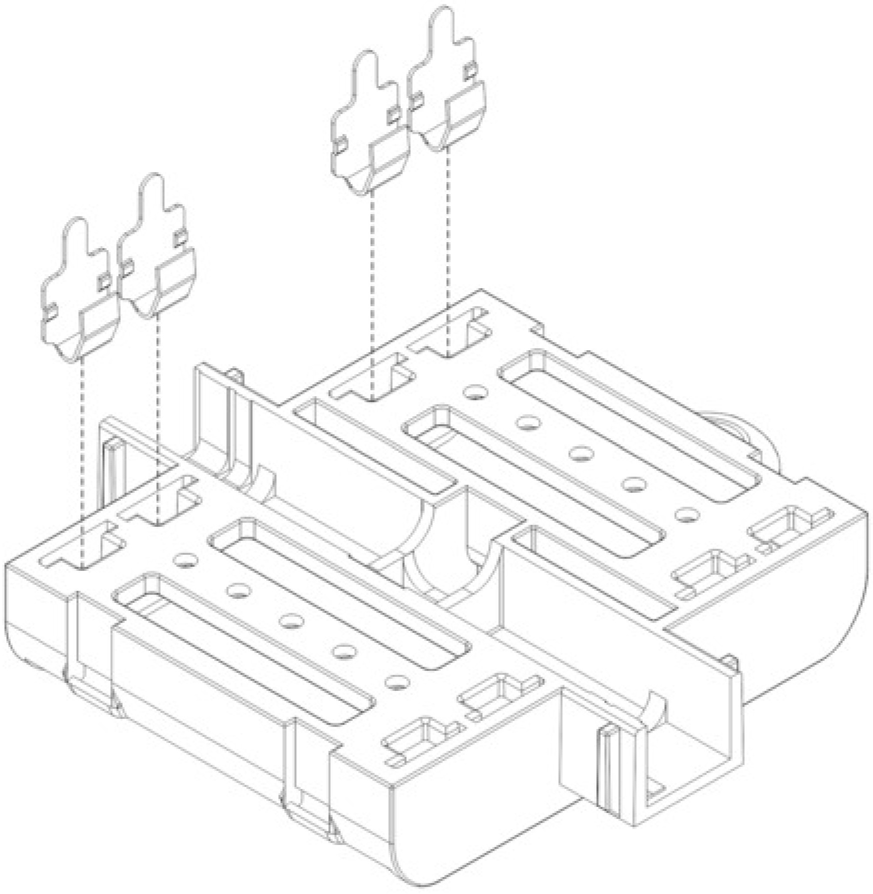
1. Before beginning assembly, plug the 3pi+ 32U4 control board in to USB and verify that it works by observing the behavior of its LEDs. You should see the yellow, green, and red user LEDs light briefly and turn off, and then the green user LED should start blinking slowly. Disconnect the control board after you confirm it is working.

Battery contacts

2. Insert the two double battery contacts into the <u>bottom</u> of the chassis as shown, making sure to put them on the correct side.

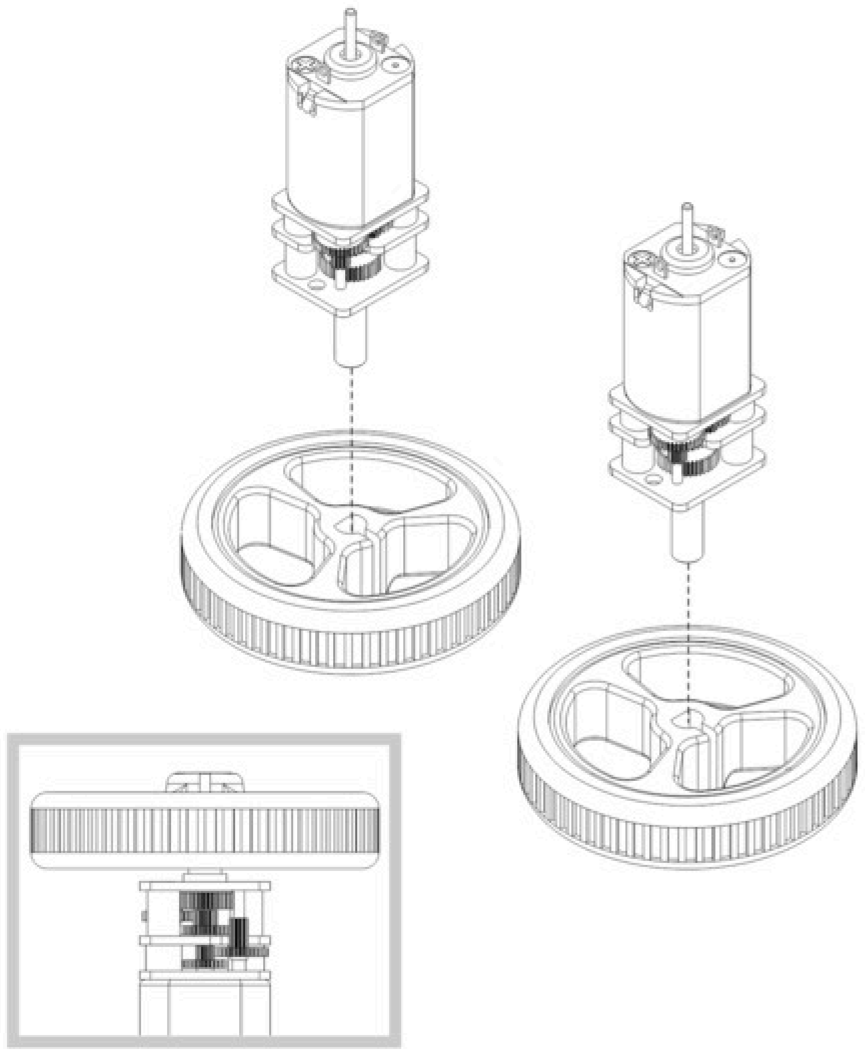


3. Insert the four single battery contacts into the top of the chassis as shown. Adjust them until they are centered, straight, and match the height of the double battery contacts.



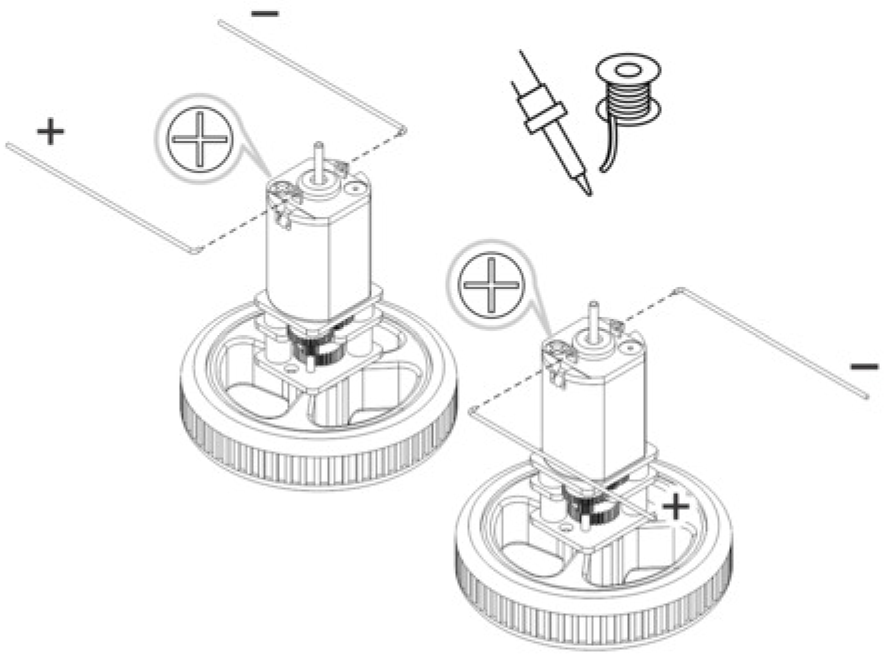
Motors and encoders

4. After installing the tires on the wheels, press the output shaft of each motor into a wheel, with the flat side of the wheel hub facing the motor. The end of the gearbox shaft should end up flush with the outside of the wheel. A good way to do this is to set the wheel on a flat surface (like a table top) and press the motor shaft into the wheel until it contacts the surface.



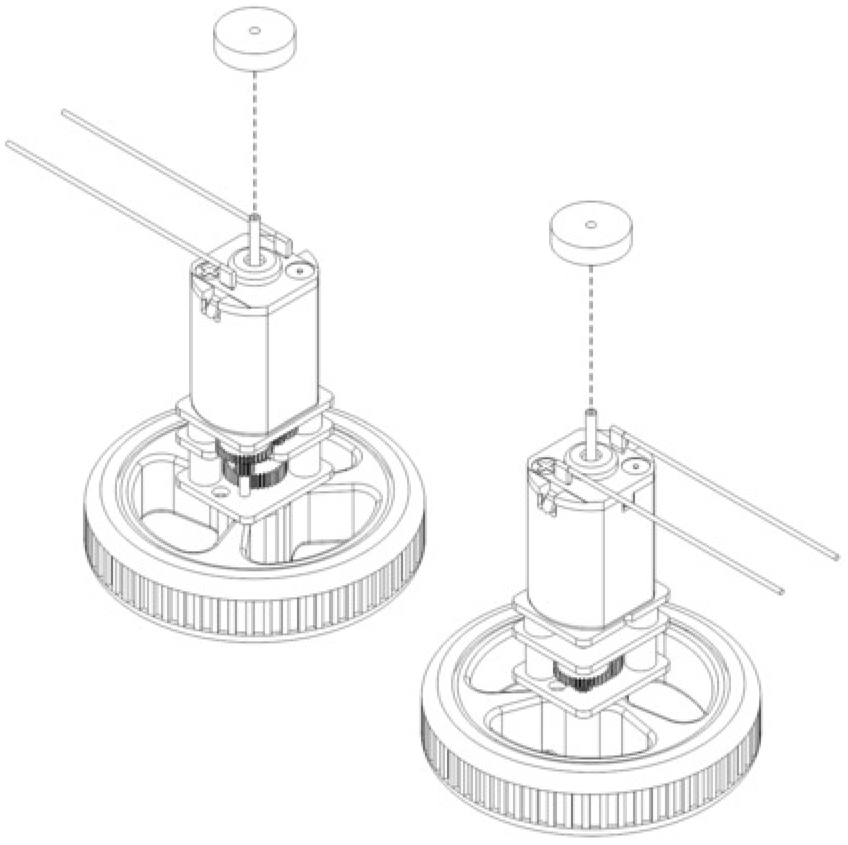
5. Cut two of the included jumper wires in half to form four segments, and trim off the ends that are covered in adhesive (the adhesive could interfere with making a good electrical connection to the motor). These wire segments will be used as motor leads.
6. Solder a pair of leads to each motor, paying attention to the way the motor will eventually be oriented in the chassis (see below). You might find it helpful to make a small bend at the tip of each lead to hook into the hole in the motor lead tab to hold it in place for soldering.

Warning: Holding the soldering iron against the motor lead for more than a few seconds can start to damage the motor brushes, so try to be reasonably quick/efficient with this soldering. If the first attempt does not go well, remove the soldering iron and let the motor cool for a few seconds before trying again.

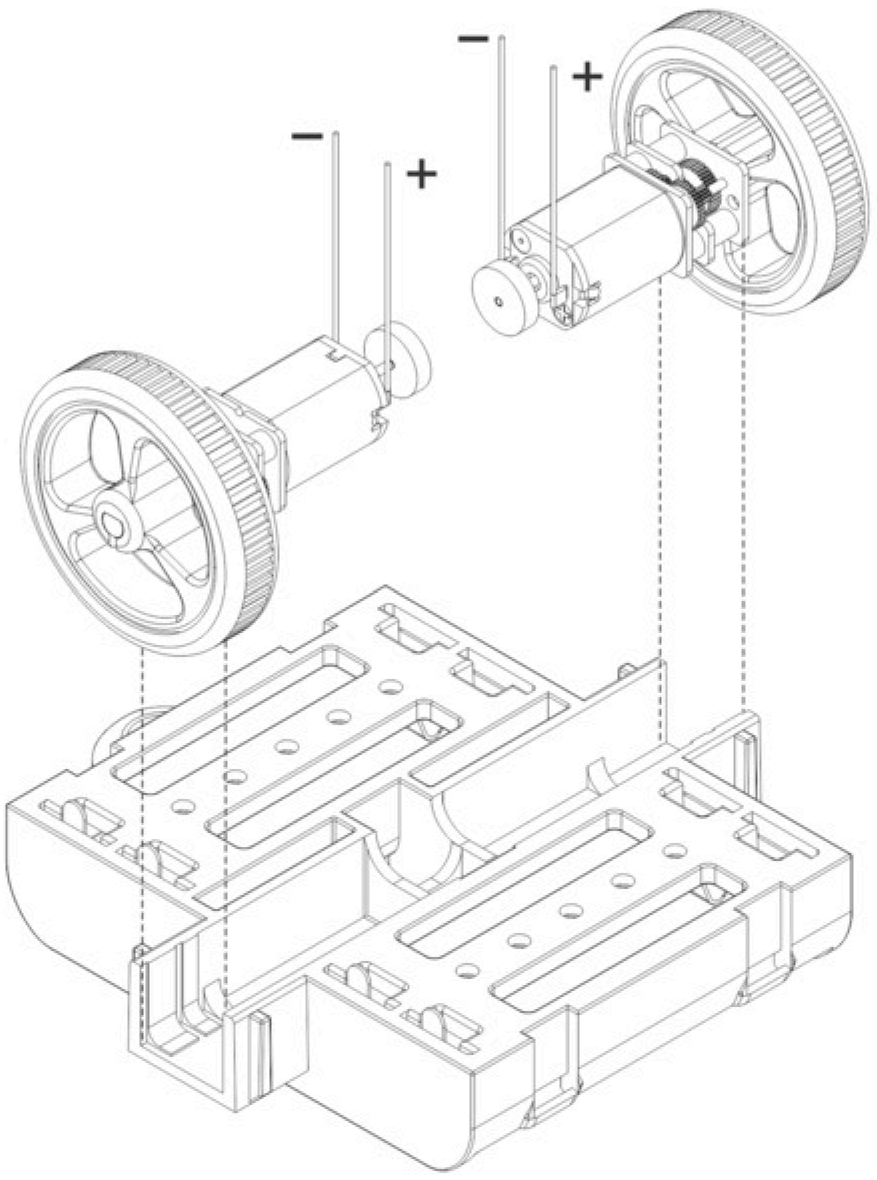


Each motor's positive terminal is indicated by a plus sign (+) in the black plastic end of the motor. For consistency, we recommend soldering the motors to the control board with the positive terminal closest to the front, so you should attach the leads to allow the motors to be oriented this way. (Don't worry if you accidentally get the orientation of one or both motors wrong, though. You can later correct for it in software with our Pololu3piPlus32U4 library.)

7. Press a magnetic encoder disc onto the motor shaft of each motor so that the end of the shaft is flush with the back of the disc. One easy way to accomplish this is to press the motor onto the disc while the disc is sitting on a flat surface, pushing until the shaft makes contact with that surface.

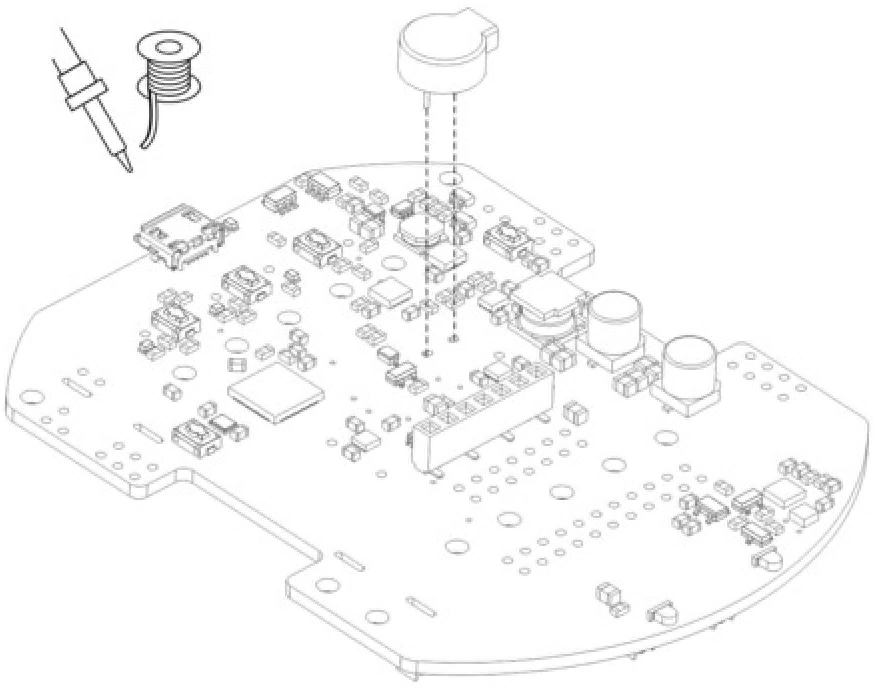


8. Place the motors into the channel in the middle of the chassis, aligning each gearbox with the grooves in the channel. The outer plate of the gearbox should be even with the edge of the chassis.

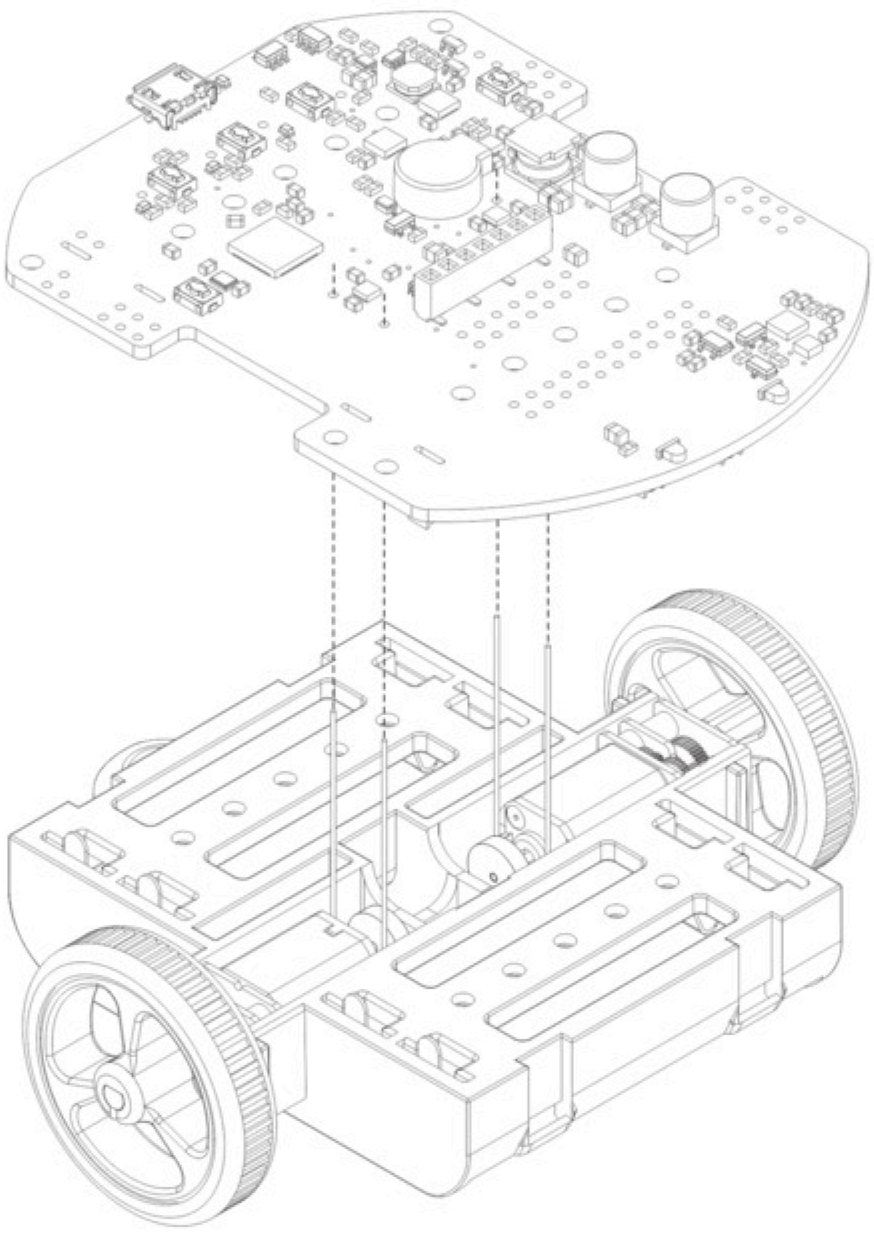


Control board and bumper skirt

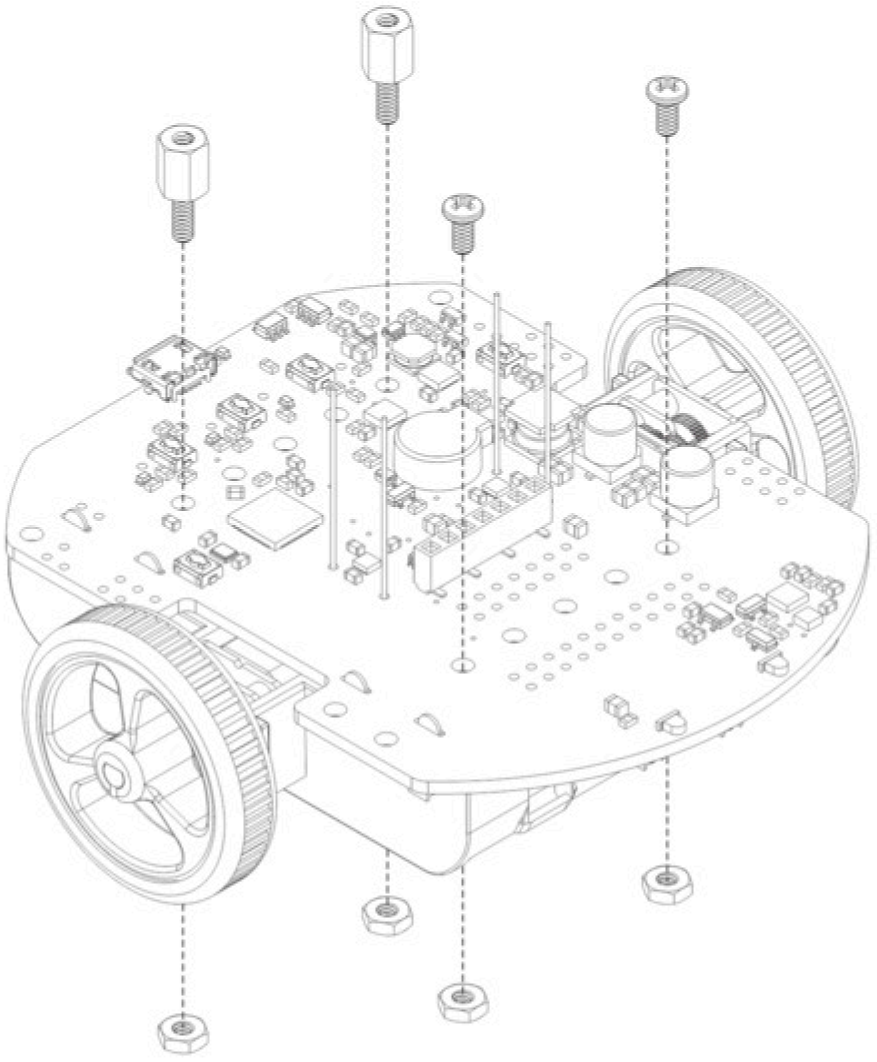
9. Solder the buzzer to the 3pi+ 32U4 control board.



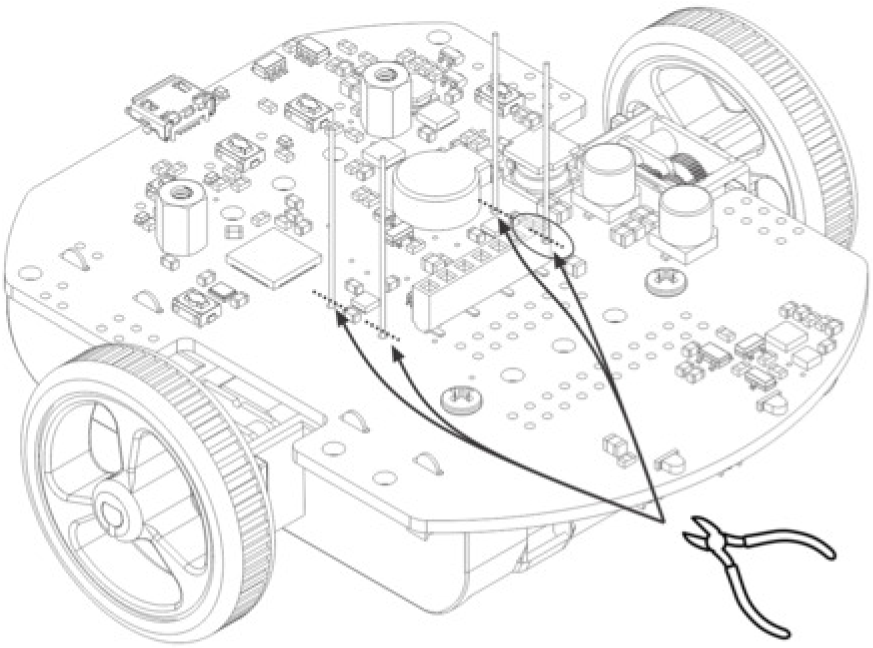
10. **Optional:** This is a convenient time to add any other optional electronics or headers.
11. Place the control board on the chassis. The motor leads and single battery contacts should be inserted into the corresponding through holes.



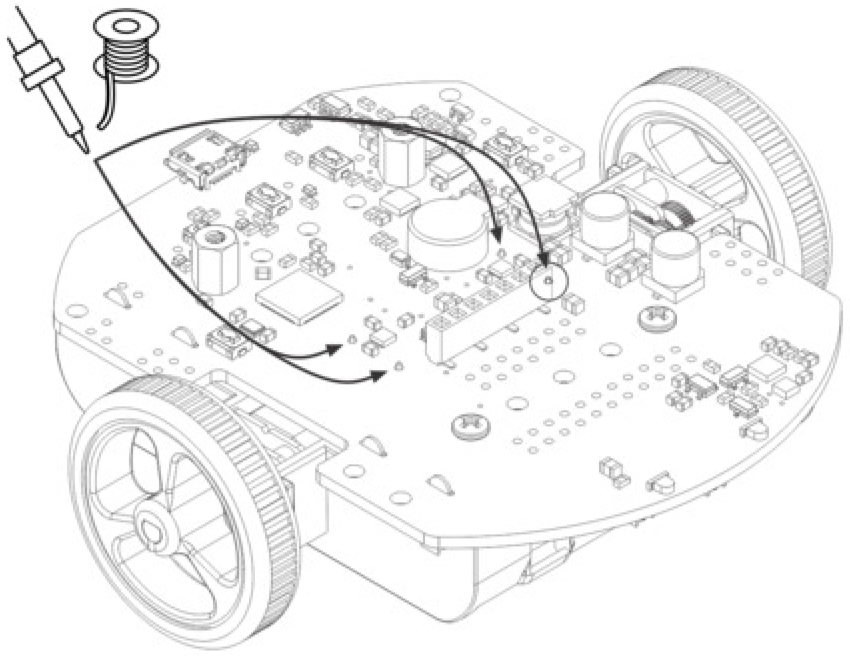
12. Screw the control board to the chassis: we recommend using two screws in the outermost holes of the front row and two standoffs in the outermost holes of the back row. (For the original LCD version, which does not include standoffs, you can use screws in all four locations instead.) In each of the four mounting holes, insert a #2-56 machine screw or standoff through the main board and chassis, and tighten it against a nut under the chassis. It is usually easier to place the nut into the recess first and hold it there with a finger or piece of tape while inserting the screw or standoff.



13. Trim off the excess length of wire from each motor lead.

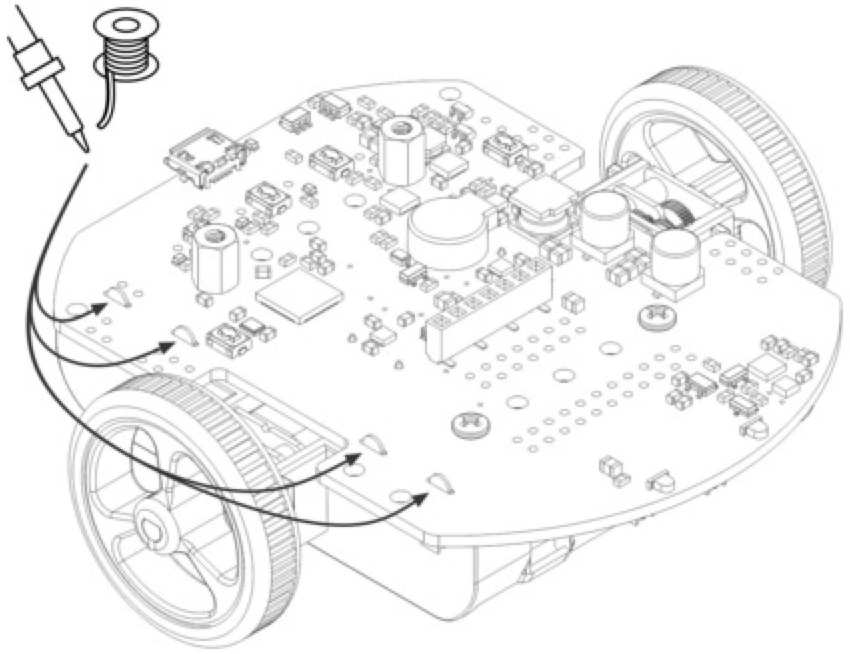


14. Solder the motor leads to the main board.

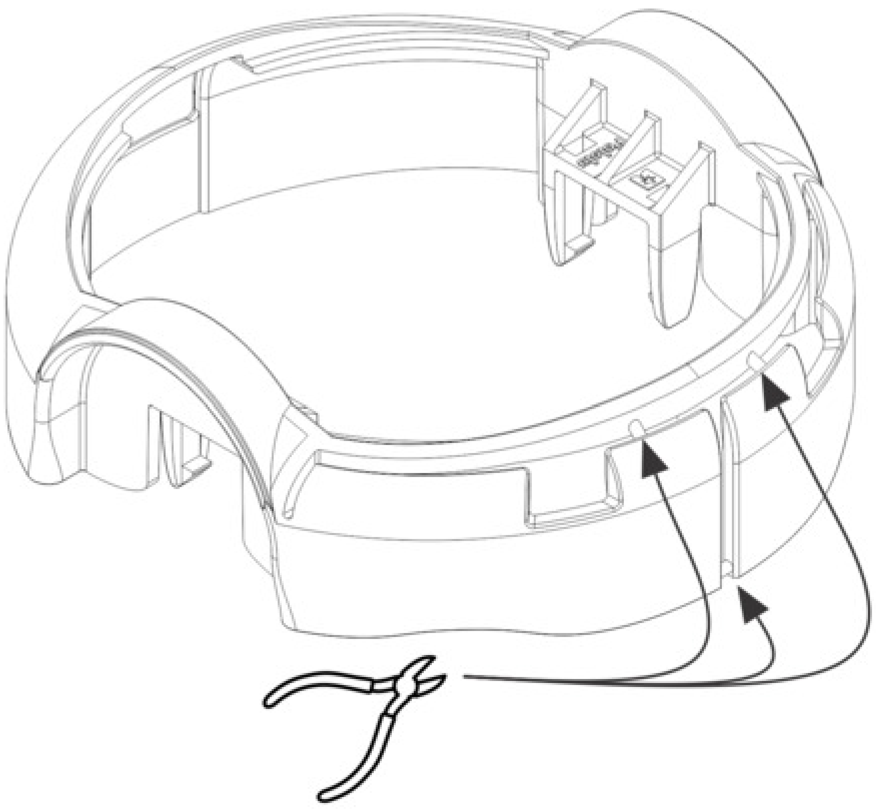


15. Double-check the alignment of the battery contacts, then solder the single battery contacts to

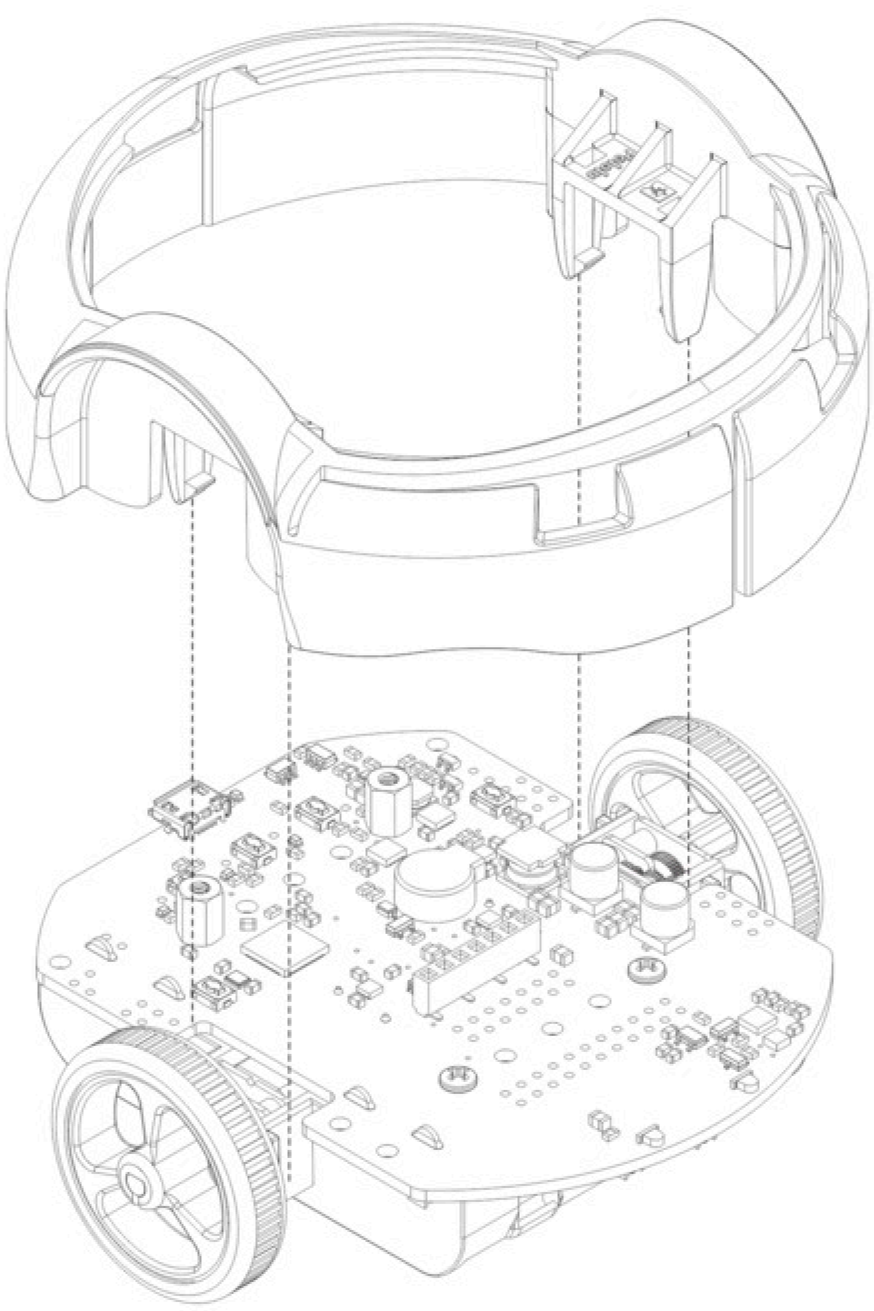
the main board.



16. **Optional:** We recommend cutting the supports for the flaps on the front of the bumper skirt, since the flaps need to be able to deflect for the bump sensors on the 3pi+ 32U4 to work. The easiest way to do this is with a pair of flush cutters, but you can also use diagonal cutters or a knife (which might leave behind bumps that you need to clean up with a file).



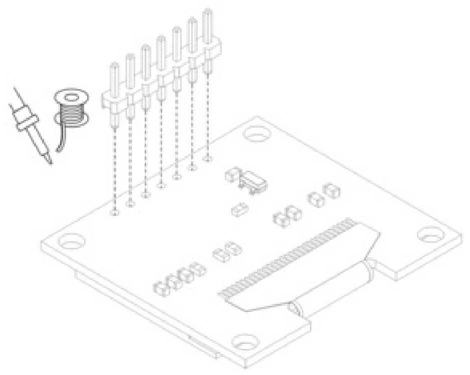
17. Install the bumper skirt by pushing the clips on each side over the motor housings until they snap into place.

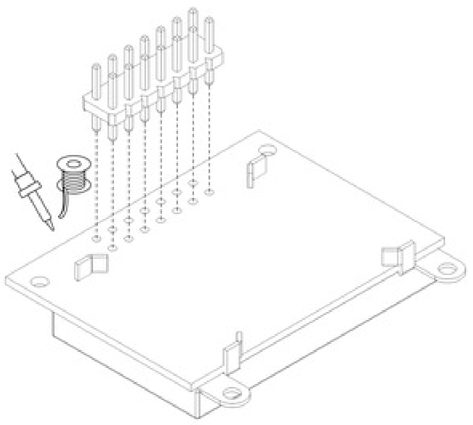


Display

18. Solder the 1×7 low-profile header to the OLED display (or the 2×7 low-profile header to the LCD). The shorter side of the header should be inserted fully through the corresponding through holes from the bottom side of the display module until the header is flush, and the solder joints should be made on the top (screen) side of the display. **Tip:** Solder a single pin first and ensure the header is flush before making any additional solder joints. If the header

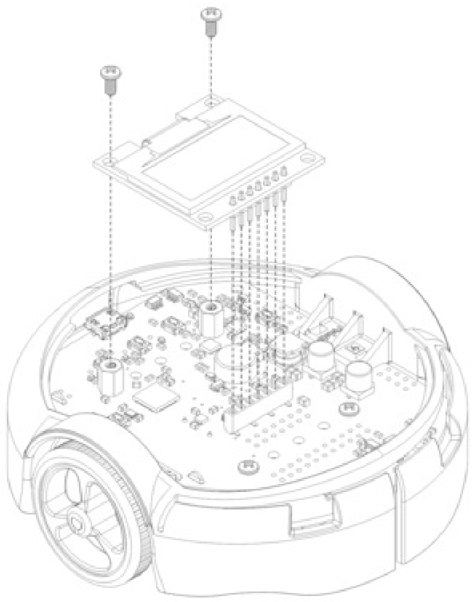
is not flush, you can use the soldering iron to re-melt the solder joint while you make the necessary adjustments. Be careful not to touch the pin you are soldering as the heat will conduct all the way through to the other end!

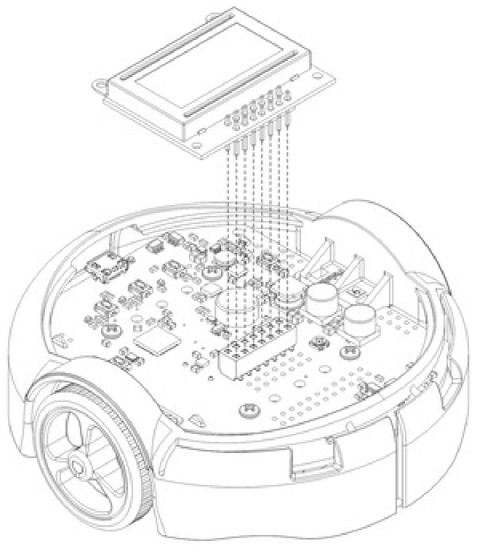

Installing header pins on the OLED display.


Installing header pins on the LCD.

19. Plug the display into the matching female header on top of the main board. You can optionally use two more #2-56 screws to secure the OLED display to the standoffs.

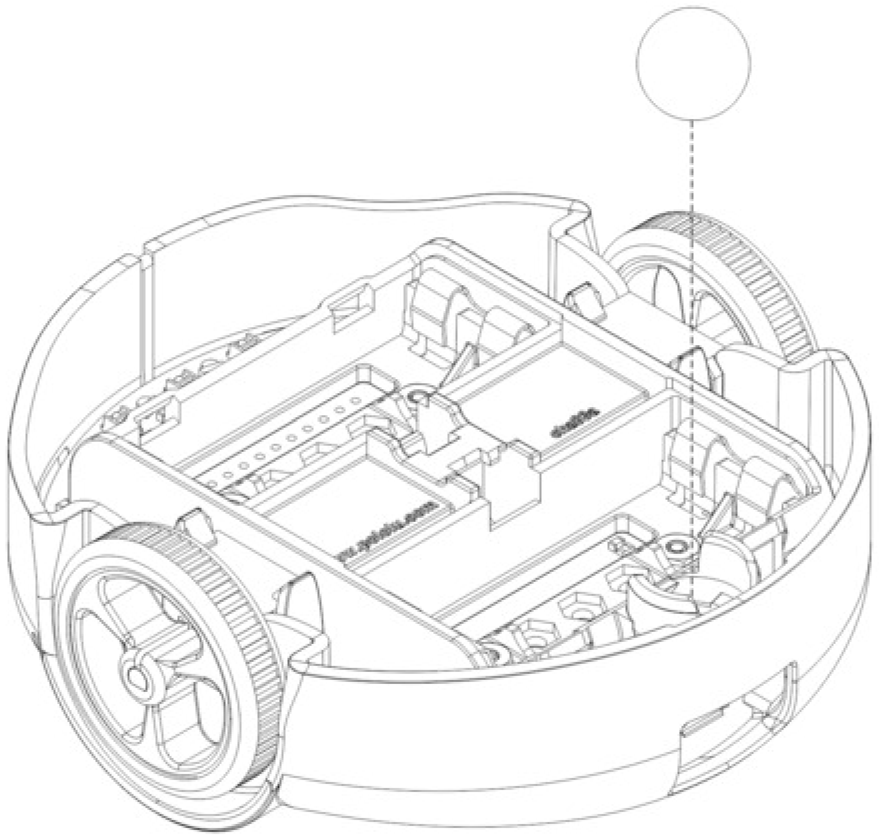
Warning: The display header does not enforce proper orientation, so it is possible to plug the display in offset or rotated 180° from its intended position. Incorrect positioning can damage the display or the control board, so please take care during this step to ensure that the display is plugged in properly.


Mounting the 3pi+ 32U4 OLED display.

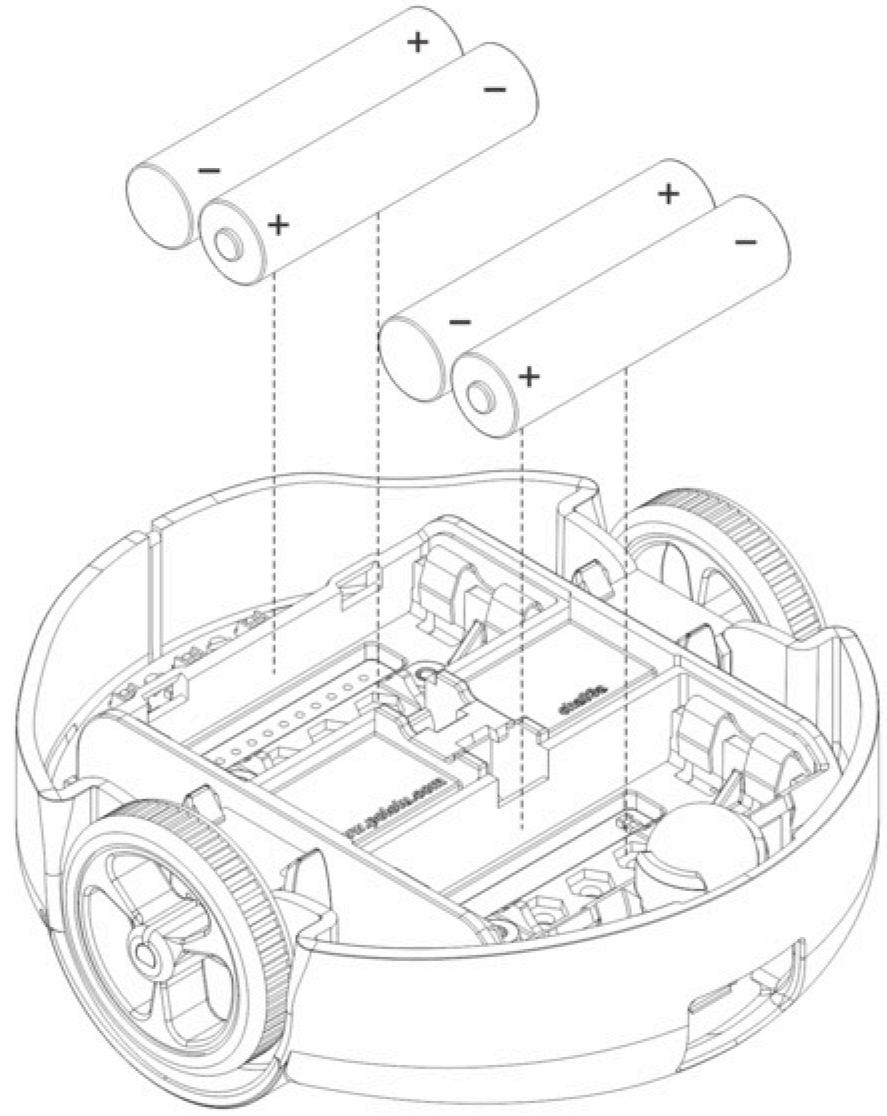

Mounting the 3pi+ 32U4 LCD.

Ball caster and battery covers

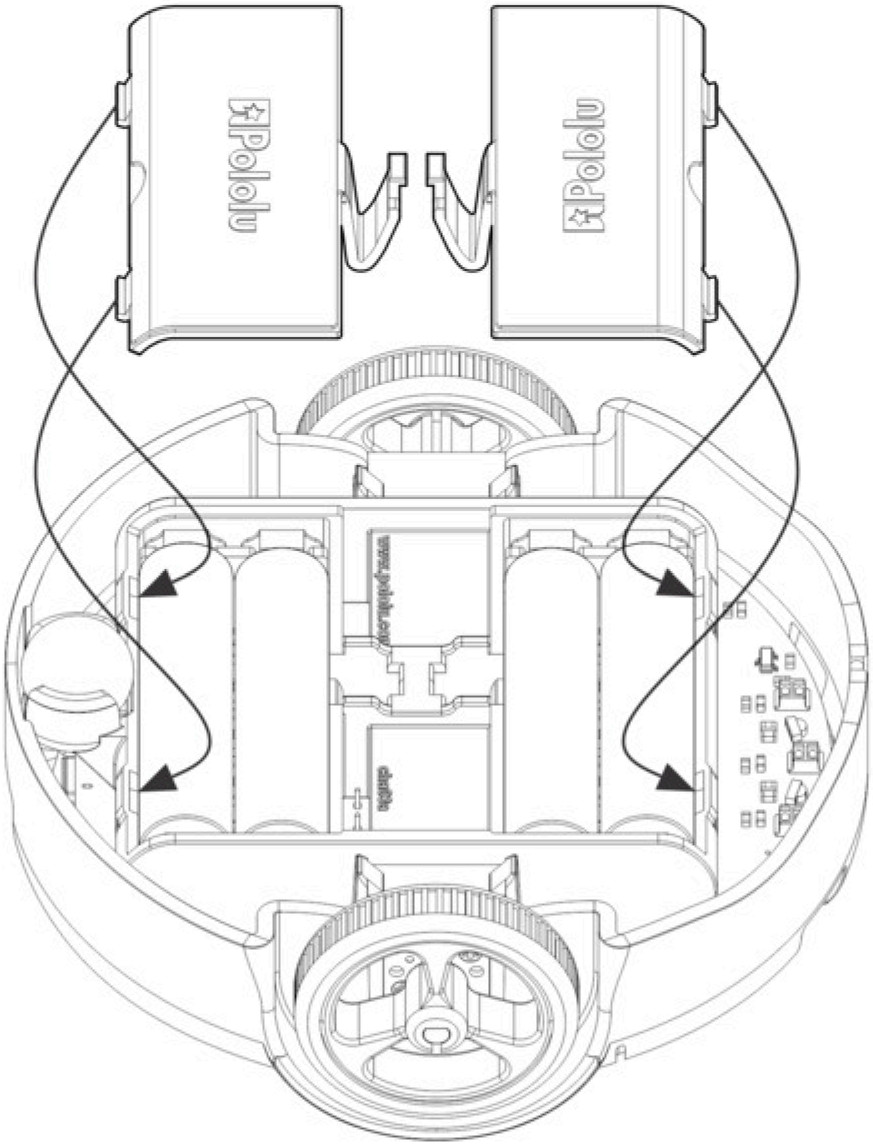
20. Press the ball caster into its socket on the bottom of the chassis.



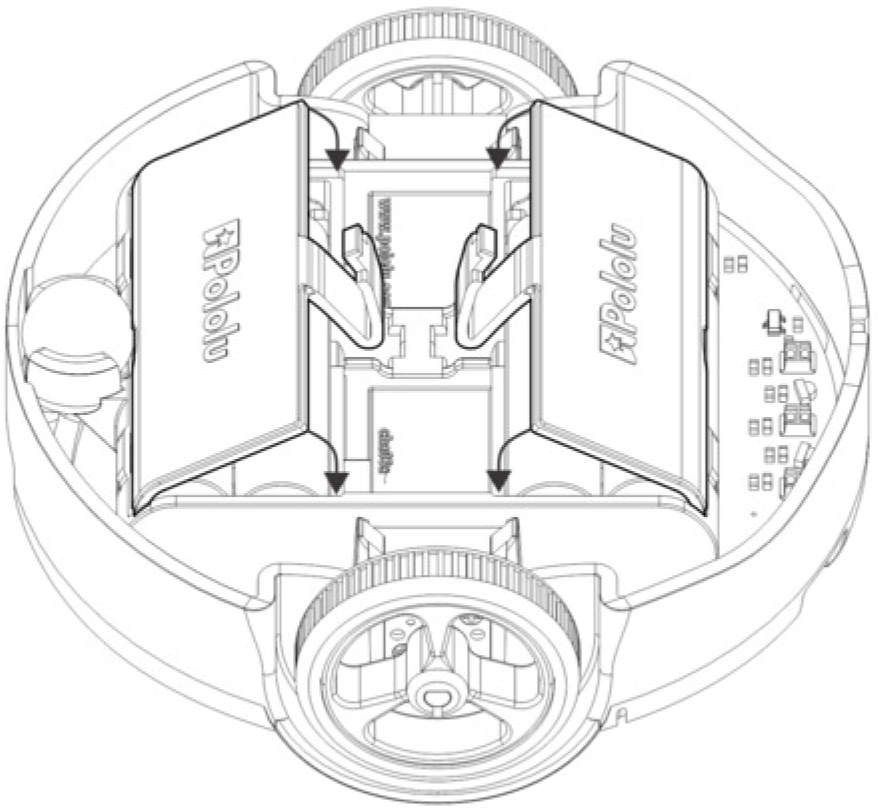
21. Install four new or freshly charged AAA batteries in the battery compartment (we recommend using rechargeable **AAA NiMH cells** **[https://www.pololu.com/product/1002]**). The correct orientation for the batteries is indicated by the silkscreen markings printed on the bottom of the control board, visible through the slots in the chassis. Be careful not to reverse any of the batteries, or else the 3pi+ 32U4 will not operate properly (although the control board will not be damaged).



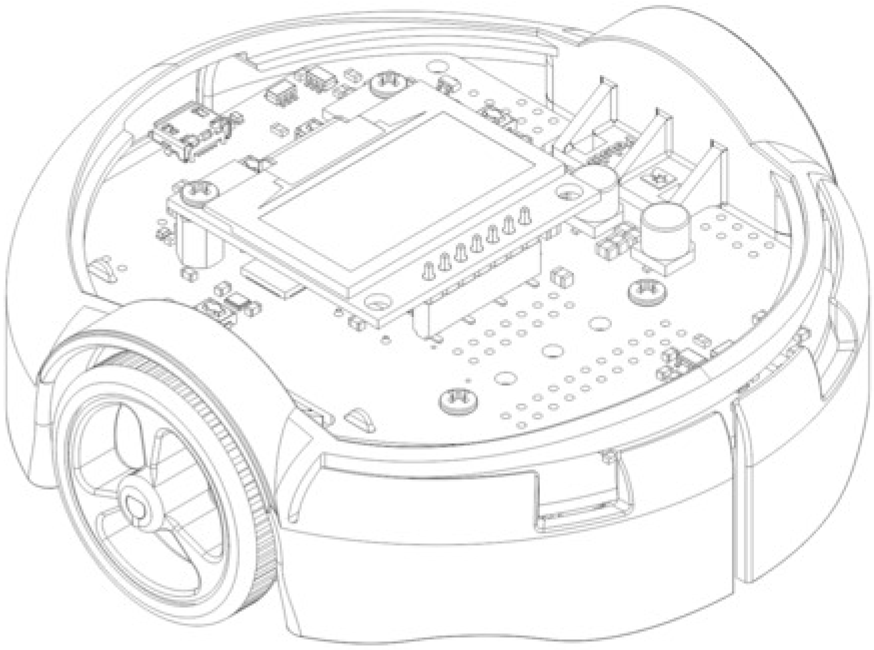
22. Secure the battery compartment covers by first hooking their tabs into the corresponding slots at the outer edges of the battery compartments…



23. …and then pivoting the covers down until the clips snap into place.



The assembly of your 3pi+ 32U4 robot is now complete, and it is ready to be used!



4. Using the preloaded demo program

Your 3pi+ 32U4 ships preloaded with a program that demonstrates most of its features. We recommend trying out this program to familiarize yourself with what it can do before moving on to programming it yourself.

When you first turn on your 3pi, you should hear a beep and see an introductory screen followed by a prompt to press the B button, which will take you to the main menu. On the original (LCD) version of the 3pi+ 32U4, if you hear a beep but do not see anything on the LCD, you might need to adjust the contrast potentiometer on the control board near the top right corner of the LCD. Once you are at the main menu, you can press C or A to scroll forward or backward through the menu, and you can press B to make a selection or to exit one of the demos. These demos are accessible from the menu:

1. **Power:** This demo displays the battery voltage in millivolts, which should be above 5000 (5.0 Volts) for a fully-charged set of rechargeable batteries or a fresh set of alkalines, and indicates whether USB power is present.
2. **LineSens:** This demo shows the current readings of the five IR line sensors using a bar graph. Bigger bars mean lower reflectance (e.g. the sensor is over something dark). Placing a reflective object such as your finger under one of the sensors should cause the corresponding reading to drop visibly on the graph. This demo also displays “C∗” or “C·” to indicate that button C has an effect on the IR emitters—the emitters are on by default, and pressing C will turn them off for as long as the button is held. In indoor lighting conditions

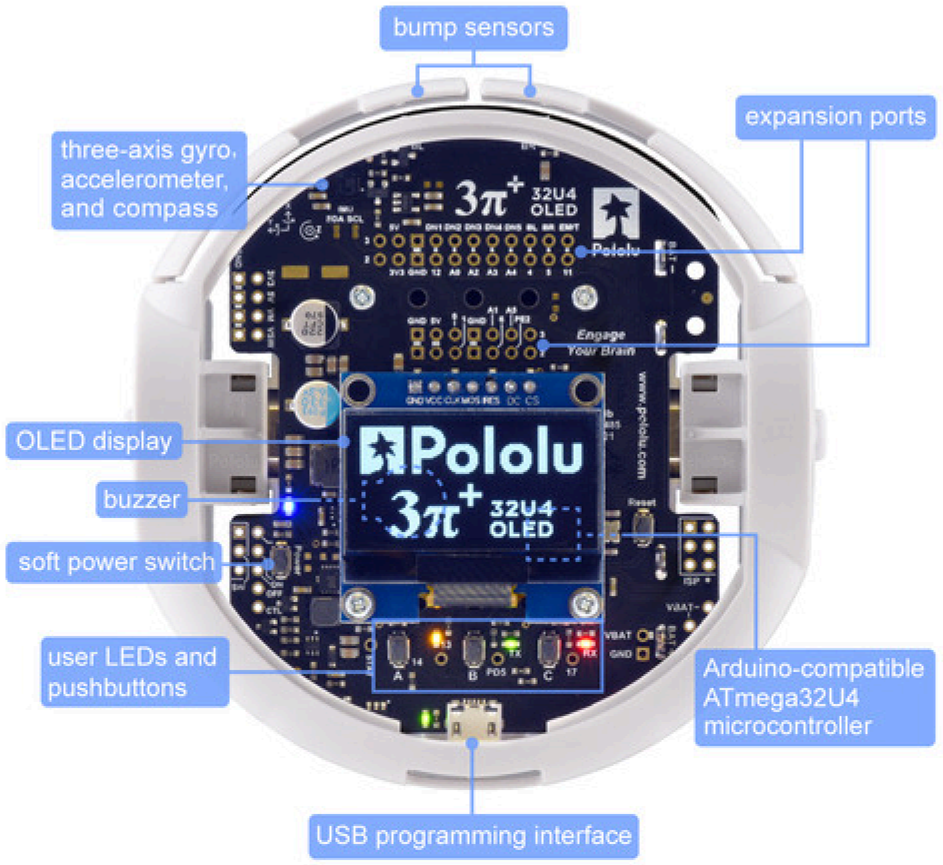
away from bright incandescent or halogen lights, all of the sensors should return entirely black readings with the IR emitters off.

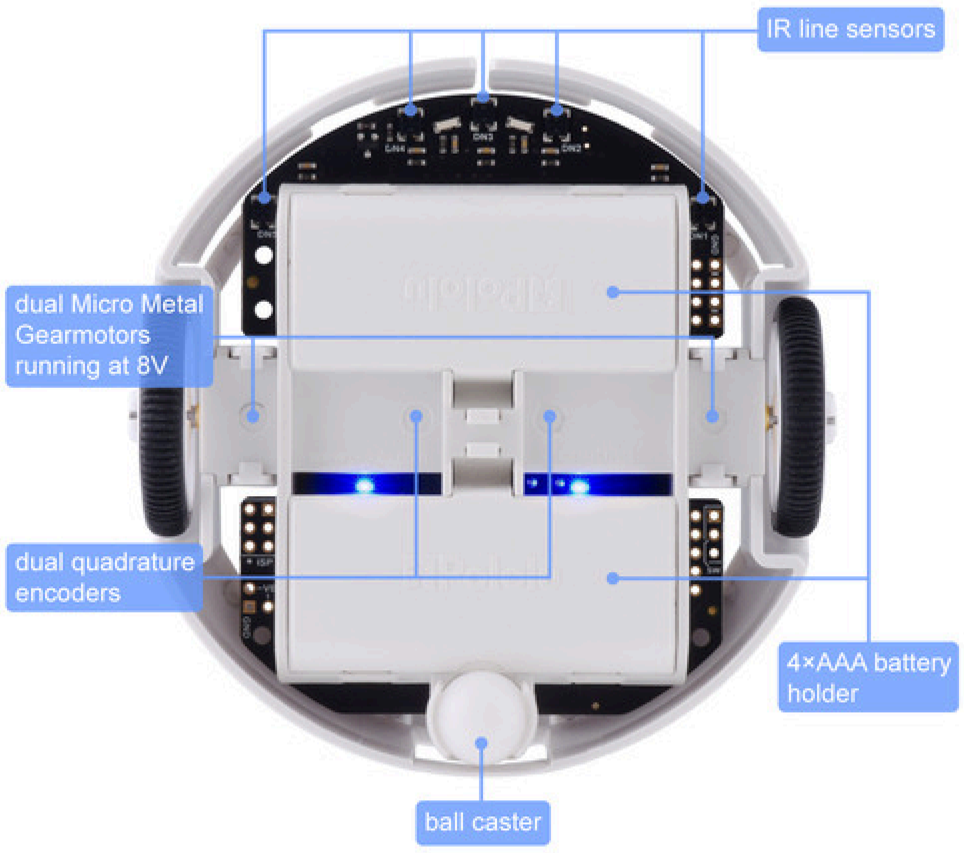
3. **BumpSens:** This demo lets you try out the bump sensors along the front side of the robot. While the left sensor is pressed, the screen will display “L” and the yellow LED will be on. While the right sensor is pressed, the screen will display “R” and the red LED will be on. The buzzer will also beep each time a bump sensors is pressed, with a different tone for left and right. ***Note:** this demo will only work if the bump sensor supports have been cut; assembled versions ship with them already cut, but you will need to cut them yourself as described in the assembly instructions (**Section 3**) if you got the kit version.*
4. **Inertial:** This demo lets you try out the robot’s inertial sensors. As you make the robot roll, pitch, or yaw in your hands, the top line of the screen will display which axis (±X, Y, or Z) the robot is rotating about, as measured by the integrated three-axis gyro. While the robot is stationary, it should just display “Rot”, with no axis listed. The bottom line of the screen will display which robot axis is currently pointing up (i.e., in the opposite direction of gravity), as measured by the integrated three-axis accelerometer. When sitting on a flat table, it should say “Up+Z”. You can try tilting various parts of the robot to face up to see how that changes.
5. **Compass:** This demo uses the robot’s integrated three-axis magnetometer to measure the Earth’s magnetic field and determine which compass direction it is facing. The demo is constantly calibrating, and in order to work well, the robot should initially be rotated about various axes (the screen will start by displaying “Turn me!” to indicate this). Please note that the Earth’s magnetic field can be distorted by stronger local magnetic fields, especially in close proximity to iron or steel, so if the demo is not working well, try changing locations. Also, please note that magnetic field orientations vary across the surface of the Earth, and magnetic north often deviates from true north.
6. **Motors:** This demo lets you try out making the motors spin. Hold down A or C to run the motor on the corresponding side, or hold down both buttons to run both motors simultaneously. The motors will gradually ramp up to speed; in your own programs, you can switch them on much more abruptly. Tap A or C to flip the rotation direction of the corresponding motor (the arrows next to the button letters on the display indicate which direction the motors will turn).
7. **Encoders:** This demo displays the left and right motor encoder counts on the screen. You can rotate the wheels by hand to see the counts increase or decrease (based on the direction of rotation). You can also use the A and C buttons to activate the motors rather than turning them by hand (they function the same way as they do in the motors demo described above).
8. **Spin:** This demo makes the 3pi spin in place in both directions. Set it on the ground, press the B button to initiate the demo, and then step back and watch it spin!
9. **LEDs:** This demo cycles through lighting the red, green, and yellow user LEDs.

10. **OLED** or **LCD:** This demo shows you various text characters you can display on the OLED or LCD screen. Use the buttons to scroll through the pages of characters.
11. **Music:** Plays an adaptation of J. S. Bach's Fugue in D Minor for microcontroller and buzzer, while scrolling a text display. This demonstrates the ability of the 3pi+ to play music in the background.
12. **About** *(original LCD version only)***:** This displays basic information about the 3pi+ on the LCD.

The source code for the demo program is included as part of the Pololu3piPlus32U4 Arduino Library (**DemoForLCDVersion** and **DemoForOLEDVersion** for the respective display types). See **Section 7** for more information.

5. The 3pi+ 32U4 in detail


bump sensors
expansion ports
three-axis gyro, accelerometer, and compass
OLED display
buzzer
soft power switch
user LEDs and pushbuttons
Arduino-compatible ATmega32U4 microcontroller
USB programming interface



5.1. Microcontroller

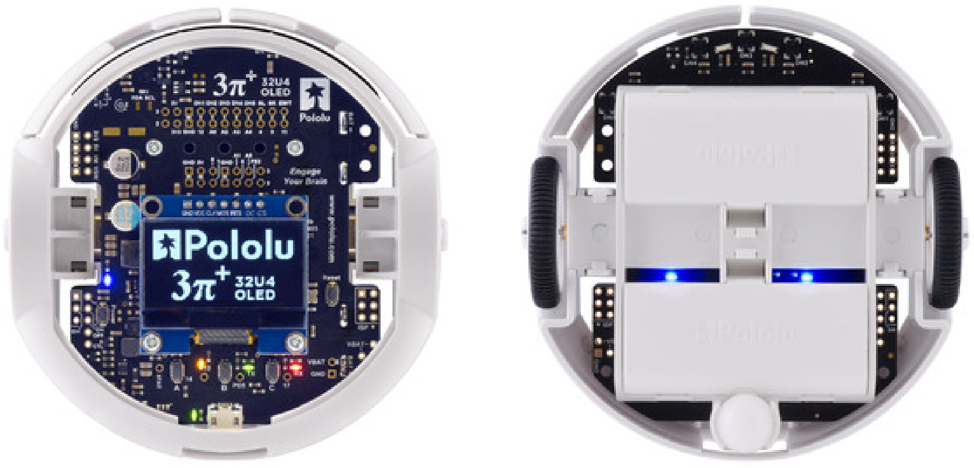
Like our **A-Star 32U4 programmable controllers [https://www.pololu.com/category/149/a-star-programmable-controllers]**, the 3pi+ 32U4 control board features an integrated, USB-enabled ATmega32U4 AVR microcontroller from Atmel, clocked by a precision 16 MHz crystal oscillator. This is the same microcontroller and clock frequency used in the **Arduino Leonardo [https://www.pololu.com/product/2192]** and **Arduino Micro [https://www.pololu.com/product/2188]**.

The control board includes a USB Micro-B connector that can be used to connect to a computer's USB port via a **USB A to Micro-B cable [https://www.pololu.com/product/2072]** (not included). The USB connection can be used to transmit and receive data from the computer and program the board over USB. The USB connection can also provide power for the microcontroller and most of the other hardware on the board (but not motor power); see **Section 5.7** for more details.

The control board's ATmega32U4 comes preloaded with the Arduino-compatible **A-Star 32U4 USB bootloader [https://www.pololu.com/docs/0J83/9]**, which allows it to be easily programmed using the Arduino IDE. For more information about programming the 3pi+ 32U4, see **Section 6**.

The board also has a location for a 6-pin ISP header that allows it to be programmed with an external programmer, such as our **USB AVR programmer [https://www.pololu.com/product/3172]**. Pin 1 of the header is indicated with a small white dot and has an octagonal shape.

5.2. User interface



LEDs

The 3pi+ 32U4 control board has seven indicator LEDs, three of which are user-controllable:

- A **yellow** user LED is connected to Arduino digital pin 13, or PC7. You can drive this pin **high** in a user program to turn this LED on. The **A-Star 32U4 Bootloader [https://www.pololu.com/docs/0J83/9]** fades this LED on and off while it is waiting for a sketch to be loaded.

- A **green** user LED is connected to Arduino pin 30, or PD5, and lights when the pin is driven **low**. While the board is running the A-Star 32U4 Bootloader or a program compiled in the Arduino environment, it will flash this LED when it is *transmitting* data via the USB connection.

- A **red** user LED is connected to Arduino pin 17, or PB0, and lights when the pin is driven **low**. While the board is running the A-Star 32U4 Bootloader or a program compiled in the Arduino environment, it will flash this LED when it is *receiving* data via the USB connection.

These three user LEDs are all located near the bottom of the board, and the Pololu3piPlus32U4 library contains functions that make it easier to control them (see **Section 7**). Some of the LED control lines are also display interface lines (green and red on the OLED version; all three LEDs on the original LCD version), so you will see them flicker when you update the display. The green and red user LEDs also share I/O lines with pushbuttons (see below).

The remaining four LEDs are power indicators:

- A **blue** power LED next to the power button indicates when the 8 V motor voltage regulator is active. The regulator is powered by the 3pi+'s batteries, so the power switching circuit must be turned on.

- A **blue** power LED on the left underside of the 3pi+ (closer to the power button) indicates when the 5 V regulator is active. The regulator is powered by the 3pi+'s batteries, so the power switching circuit must be turned on.

- A **blue** power LED on the right underside of the 3pi+ (farther from the power button) indicates when the 3pi+'s logic circuit, including the microcontroller, is receiving 5V. Logic power can come from either the 5 V regulator or from USB, so this LED will be lit when either the power switching circuit is turned on **or** when the 3pi+ is plugged in to USB.

- A **green** power LED next to the USB connector indicates when the USB bus voltage (VBUS) is present.

Pushbuttons

The 3pi+ 32U4 control board has five pushbuttons: a **power button** on the left, a **reset button** on the right, and **three user pushbuttons** located along the rear. The user pushbuttons, labeled A, B, and C, are on Arduino pin 14 (PB3), pin 30 (PD5), and pin 17 (PB0), respectively. Pressing one of these buttons pulls the associated I/O pin to ground through a resistor.

The three buttons' I/O lines are also used for other purposes: pin 14 is MISO on the hardware SPI interface, pin 30 and pin 17 control the green and red user LEDs, and some pins are display interface lines (pin 30 and pin 17 on the OLED version; all three buttons on the original LCD version). Although these uses require the pins to be driven by the AVR (or SPI slave devices in the case of MISO), resistors in the button circuits ensure that the 3pi+ 32U4 Control Board will not be damaged even if the corresponding buttons are pressed at the same time, nor will SPI or display communications be disrupted. The functions in the Pololu3piPlus32U4 library take care of configuring the pins, reading and debouncing the buttons, and restoring the pins to their original states.

Buzzer

The **buzzer** **[https://www.pololu.com/product/1484]** included with the 3pi+ 32U4 control board can be soldered into the designated through-holes and used to generate simple sounds and music. By default, it is connected to digital pin 6 (which also serves as OC4D, a hardware PWM output from the AVR's 10-bit Timer4). If you alternate between driving the buzzer pin high and low at a given frequency, the buzzer will produce sound at that frequency. You can play notes and music with the buzzer using functions in the Pololu3piPlus32U4 library. If you want to use pin 6 for an alternate purpose, you can disconnect the buzzer circuit by cutting the surface-mount jumper next to the buzzer.

Display header

The 3pi+ 32U4 OLED control board has a 1×7 header where you can connect a **graphical OLED module [https://www.pololu.com/product/3760]** with a low-profile male header. The included display has a resolution of 128×64 pixels and uses an **SH1106 controller [https://www.pololu.com/file/0J1813/SH1106.pdf]** (1MB pdf), which the 3pi+ communicates with via software SPI. On-board level shifters convert 5 V signals from the 3pi+'s microcontroller to the 3.3 V logic level required by the OLED module.

The original 3pi+ 32U4 control board has a 2×7 header where you can connect an **8×2 character LCD [https://www.pololu.com/product/356]** with a **low-profile male header [https://www.pololu.com/product/2663]** (or any other LCD with the common **HD44780 parallel interface [https://www.pololu.com/file/0J71/DMC50448N-AAE-AD.pdf]** (109k pdf)). You can adjust the LCD contrast with the potentiometer directly above the LCD connector. We recommend using a 2 mm slotted screwdriver to adjust the contrast.

The Pololu3piPlus32U4 library provides functions to show data on a connected display. It is designed to gracefully handle alternate use of the display interface lines by only changing pin states when needed for a display command, after which it will restore them to their previous states. This allows the display interface lines to be used for other functions (such as pushbutton inputs and LED drivers).

Although the OLED and LCD screens have different hardware interfaces, the library presents similar software interfaces for both that generally allow code written for the original (LCD) version of the 3pi+ 32U4 to work on the OLED version with minimal changes. The reverse is also true as long as your code does not make use of the OLED's graphical capabilities.

5.3. Motors

The 3pi+ 32U4 kit and robot are with available three different motor options:

3pi+ 32U4 Version	Micro Metal Gearmotor	No-Load Performance at 6 V	Stall Extrapolation at 6 V	Top 3pi+ Speed
Standard Edition	30:1 MP 6V	720 RPM, 40 mA	0.33 kg·cm, 0.67 A	1.5 m/s
Turtle Edition	75:1 LP 6V	180 RPM, 20 mA	0.64 kg·cm, 0.36 A	0.4 m/s
Hyper Edition	15:1 HPCB 6V	2100 RPM, 100 mA	0.25 kg·cm, 1.5 A	~4 m/s

The hyper edition is very difficult to control and fast enough to damage itself from impacts, so it is only recommended for advanced users. **We strongly recommend keeping motor speeds below 50% on this version.** A spare gearbox is included with this edition and instructions for installing it can be found **here [https://www.youtube.com/watch?v=tQbwaJjkHVs]**.

You can also assemble the 3pi+ chassis and 3pi+ 32U4 Control Board with **different motor and gear**

ratio combinations [https://www.pololu.com/category/141/micro-metal-gearmotors-with-extended-motor-shafts] to make your own custom 3pi+ 32U4 robot. Please keep in mind that using faster or lower-torque motors will make your robot more difficult to control.

Two on-board motor drivers power the 3pi+ 32U4's two Micro Metal Gearmotors. Four Arduino pins are used to control the drivers:

- **Digital pin 15**, or PB1, controls the **right motor direction**.
- **Digital pin 16**, or PB2, controls the **left motor direction**.
- **Digital pin 9**, or PB5, controls the **right motor speed** with PWM (pulse width modulation) generated by the ATmega32U4's Timer1.
- **Digital pin 10**, or PB6, controls the **left motor speed** with PWM generated by the ATmega32U4's Timer1.

The Pololu3piPlus32U4 library provides functions that allow you to easily control the motors, and it can optionally take care of flipping a direction signal for you if you accidentally soldered in a motor backwards or are using a gear ratio with an odd number of stages, so the output turns the opposite direction from the input (see **Section 7**).

The 15:1 motors used on the **Hyper Edition** of the 3pi+ 32U4 robot have gearbox output shafts that rotate in the **opposite direction** from the motors' pinion gears (unlike the motors used in the other editions, where the directions are the same). For consistency, we install the motors with the positive terminal forward on all assembled 3pi+ 32U4 robots, and we recommend building kits the same way.

This means the same inputs will produce different motor directions on a Hyper Edition robot compared to a non-Hyper robot, so a program written for one might need to be modified to work well on the other (by using the direction flipping functions provided by the Pololu3piPlus32U4 library, for example).

Batteries and motor performance

As your batteries discharge, the voltage they supply will decrease. However, since the 3pi+ 32U4 uses a regulated motor voltage (see **Section 5.7** for more details), battery voltage does not typically have a major impact on the performance of the motors; they will be powered with 8 V as long as the motor voltage regulator is operating normally.

Even with a regulated motor voltage, the condition of the batteries starts to matter more as the motors draw more current (such as when accelerating, reversing, or stalled). The increased current draw of the motors causes the regulator to draw more current from the batteries in turn, and if this causes the battery voltage to drop below the regulator’s cutoff voltage, the regulator will turn off and stop powering the motors.

With the motor voltage regulator no longer drawing current, the battery voltage usually recovers a little, at which point the regulator is able to turn on again. As it starts drawing a high current to power the motors once more, the regulator enters a cycle of repeatedly turning off and back on many times a second, which effectively results in a kind of current limiting for the motors.

This behavior occurs more often with batteries that are drained than with freshly charged batteries, which means you might notice the performance of a 3pi+ 32U4 decreasing as its batteries start to run out. (For example, it might accelerate more slowly or even be unable to reach as high of a top speed.) It can also come into play with older or lower-quality batteries, which tend to have higher internal resistances that cause more significant voltage drops. The Hyper Edition of the 3pi+ 32U4 is most likely to be affected due to the greater current demanded by its high-power motors.

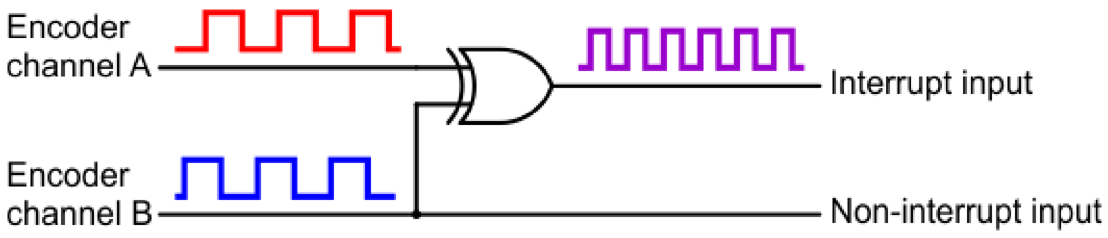
5.4. Quadrature encoders

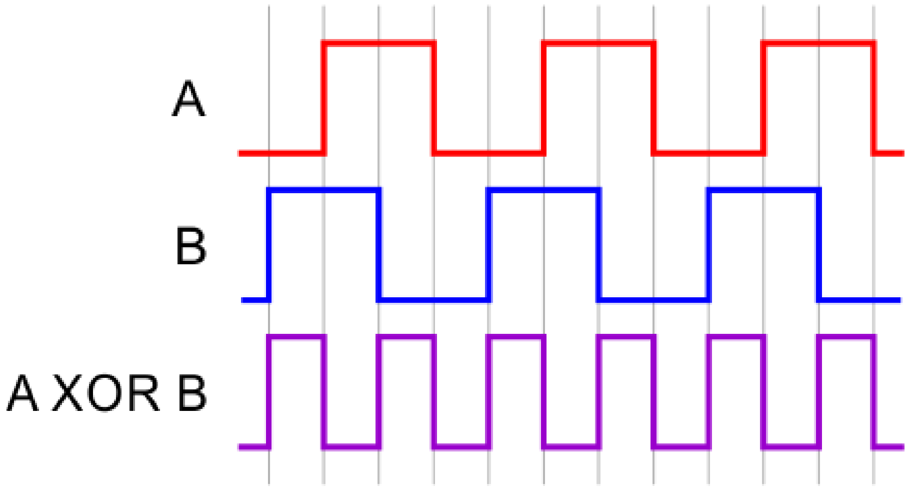
Each drive motor on the 3pi+ 32U4 has a corresponding quadrature encoder system consisting of a magnetic disc attached to the extended motor shaft and a pair of Hall effect sensors mounted on the control board. Other than the sensor orientation, these encoders work similarly to our **magnetic encoder kits for Micro Metal Gearmotors [https://www.pololu.com/product/3081]**. They can be used to track the rotational speed and direction of the robot’s wheels.

The encoders provide a resolution of 12 counts per revolution of the motor shaft when counting both edges of both channels. To compute the counts per revolution of the wheels, multiply the gearboxes’ gear ratios by 12. For example, if **30:1 motors [https://www.pololu.com/product/2364]** (which have gear ratios more accurately specified as 29.86:1) are used, the encoders provide 29.86 × 12 ≈ **358.3 CPR**. The exact gear ratios of our Micro Metal Gearmotors are specified on their product pages.

Quadrature encoder transitions are often detected by monitoring both encoder channels directly. However, since transitions on the 3pi+’s encoders can occur at high frequencies (several thousand per second) when its motors are running, it is necessary to use the AVR’s pin change interrupts or external interrupts to read the encoders. To reduce the required number of interrupt pins, the 3pi+ 32U4 control board XORs together both channels of each encoder and connects the resulting signal to an interrupt pin, while channel B of each encoder is connected to a non-interrupt pin:

- **Digital pin 7**, or PE6, reads the **right encoder XORed signal** using external interrupt INT6.
- **Digital pin 8**, or PB4, reads the **left encoder XORed signal** using pin change interrupt PCINT4.
- **Digital pin 23** (analog pin 5), or PF0, reads the **right encoder channel B**.
- Pin **PE2** reads the **left encoder channel B**.





The XORed signal and the channel B signal can be used to reconstruct the channel A signal by simply XORing them again: **(A XOR B) XOR B = A**. For both encoders on the Turtle Edition and Standard Edition, channel B leads channel A when the motor is rotating in the forward direction; that is, B rises before A rises and B falls before A falls. (The waveforms in the diagram above would be produced by forward rotation.)

Since the motor directions on the Hyper Edition are reversed (see **Section 5.3**), the behavior described above corresponds to backward rotation instead on that version.

The Pololu3piPlus32U4 library provides appropriate interrupt service routines and functions for reading the encoders and keeping track of their counts (see **Section 7**).

5.5. Line and bump sensors

The 3pi+ 32U4 features five downward-facing line sensors and two forward-facing bump sensors.

The five **line sensors** are on the underside of the board along the front edge and can help the 3pi+ distinguish between light and dark surfaces. Each reflectance sensor consists of a down-facing infrared (IR) emitter LED paired with a phototransistor that can detect reflected infrared light from the LED. The reflectance sensors operate on the same principles as our RC-type **QTR reflectance sensors** **[https://www.pololu.com/category/123/pololu-qtr-reflectance-sensors]**: the AVR uses an I/O line to drive the sensor output high, and then measures the time for the output voltage to decay. You can read more about the operating principles of these sensors in our **QTR Reflectance Sensor Application Note** **[https://www.pololu.com/docs/0J13]**.

The five line sensors are numbered 1 through 5, with line sensor 1 being the robot's left-most sensor. In the schematics, they are referred to as DOWN1, DOWN2, DOWN3, DOWN4, and DOWN5. On the control board, their signals are labeled DN1, DN2, DN3, DN4, and DN5.

The two **bump sensors** are also reflectance sensors, but rather than providing simple reflectance readings, these are designed to measure changes in reflected light as the corresponding bump sensor flaps on the front of the 3pi+'s bumper skirt are pressed (deflected). This allows the 3pi+ to detect when it has contacted another object in front of it and determine which side the contact is on. The left and right bump sensors' signals are labeled BUMPL and BUMPR in the schematics and BL and BR on the control board.

Each sensor output is protected by a 220 Ohm resistor to help prevent short circuits when the AVR is driving the corresponding I/O line.

The infrared emitters for both the line and bump sensors are controlled by the same pin, labeled EMIT. Driving this pin high illuminates the line sensor emitters, while driving it low illuminates the bump sensor emitters. When EMIT is not driven, as happens if the connected AVR pin is set to be an input, both sets of emitters will effectively be off (a small amount of current will still flow through them, but it is well under 1 mA).

The Pololu3piPlus32U4 library provides functions to help with reading the line sensors and bump sensors, and it handles control of the emitters appropriately (see **Section 7**).

Ambient light considerations

Since the line sensors and bump sensors rely on measurements of reflected infrared light, they are strongly affected by ambient sources of IR light in the surrounding environment (e.g. sunlight or strong incandescent lighting).

You can help compensate for ambient IR light by incorporating some calibration procedures in your programs. For example, the line sensors can measure the reflectance of light and dark surfaces during calibration and then report subsequent readings relative to this range, while the bump sensors can take baseline readings with the skirt flaps in their unpressed positions and then detect presses based on differences from the baselines. The Pololu3piPlus32U4 library provides support for calibrating both types of sensors in these ways.

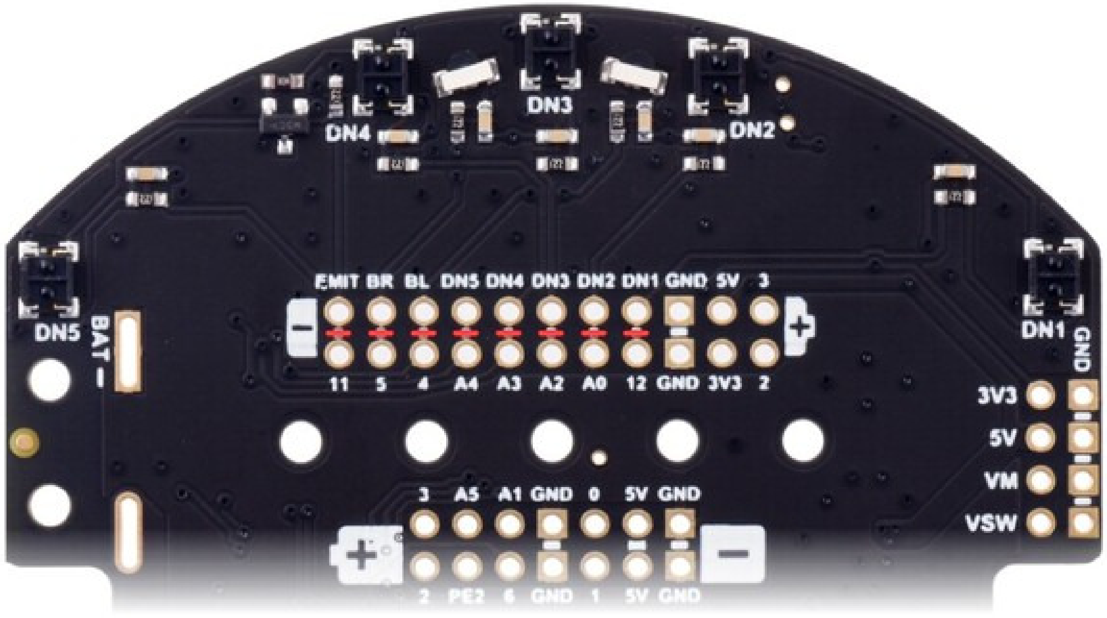
However, this calibration is not foolproof; too much ambient infrared light can still prevent the line sensors and bump sensors from working well, and the calibration will not remain effective if the ambient light level changes, such as if the 3pi+ moves from a brightly lit area to a more shaded area. The bump sensors can be especially susceptible to spurious or missed detections since they work by detecting changes in the IR light intensity.

Pin assignments and remapping

By default, the line and bump sensors support these pin assignments:

- Pin 12 is connected to line sensor 1 (DN1).
- Pin A0 (18) is connected to line sensor 2 (DN3).
- Pin A2 (20) is connected to line sensor 3 (DN3).
- Pin A3 (21) is connected to line sensor 4 (DN4).
- Pin A4 (22) is connected to line sensor 5 (DN5).
- Pin 4 is connected to the left bump sensor (BL).
- Pin 5 is connected to the right bump sensor (BR).
- Pin 11 is connected to the shared emitter control pin (EMIT).

These connections are made through traces connecting pairs of through-holes in the front expansion header of the 3pi+ 32U4 control board. A connection can be remapped by cutting the corresponding trace on the underside of the board and making a new connection between the sensor signal and another AVR pin of your choice.



Bottom view of the 3pi+ 32U4 Control Board, showing cuttable traces for remapping sensors.

5.6. Inertial sensors

The 3pi+ 32U4 includes on-board inertial sensors that allow it to determine its own orientation by implementing an inertial measurement unit (IMU). The first chip, an ST **LSM6DS33** **[https://www.pololu.com/product/2736]**, combines a 3-axis accelerometer and 3-axis gyro into a single package. The second chip is an ST **LIS3MDL** **[https://www.pololu.com/product/2737]** 3-axis magnetometer.

Level shifters built into the main board allow the inertial sensors, which operate at 3.3 V, to be connected to the ATmega32U4 (operating at 5 V). The sensors, level shifters, and I²C pull-up resistors are connected to the SDA (digital pin 2, or PD1) and SCL (digital pin 3, or PD0) pins on the AVR by default, but they can be disconnected by cutting the surface-mount jumpers labeled “SDA” and “SCL” under “IMU” on the board to allow those pins to be used for other purposes.

We recommend carefully reading the datasheets listed above to understand how these sensors work and how to use them.

Using the sensors

The Pololu3piPlus32U4 library (see **Section 7**) includes functions that help configure and read the inertial sensors. The library includes some example programs that demonstrate how to use the sensors.

For advanced applications, you can instead use some of the dedicated libraries that we have written for these sensor chips; these include our **LSM6 Arduino library** **[https://github.com/pololu/lsm6-arduino]**, **LIS3MDL Arduino library** **[https://github.com/pololu/lis3mdl-arduino]**, **LSM303 Arduino library** **[https://github.com/pololu/lsm303-arduino]**, and **L3G Arduino library** **[https://github.com/pololu/l3g-arduino]**. The 3pi+ 32U4 control board uses the same inertial sensor ICs as our **MinIMU-9 v5** **[https://www.pololu.com/product/2738]**, so Arduino software written for the MinIMU-9 (such as our **AHRS example** **[https://github.com/pololu/minimu-9-ahrs-arduino]**) can also be adapted to work on a 3pi+ 32U4.

Notes on the magnetometer

Please note that the magnetometer on the 3pi+ 32U4 can be affected by magnetic fields from the 3pi+ itself. These include magnets in the motors and encoders, electrical currents through the board, and hard iron distortions from metal (probably mostly from the batteries). The magnetometer is positioned as far away from the motors as possible to avoid interference from them, but hard iron distortions can still influence the readings significantly, making it difficult to accurately determine the 3pi+’s absolute heading based on the raw magnetometer data.

This **post on the Pololu forum** **[https://forum.pololu.com/t/correcting-the-balboa-magnetometer/14315]** details a technique for correcting for hard iron distortions, making it possible to use of the magnetometer as a compass for navigation in environments that are not dominated by magnetic interference. (It is written about our **Balboa 32U4 robot** **[https://www.pololu.com/category/210/balboa-robot-and-accessories]**, but the same principles apply to the 3pi+.)

5.7. Power

The 3pi+ 32U4 control board includes battery terminal connections that provide access to power from the 3pi+ chassis’s four-AAA battery compartment. We recommend using rechargeable AAA NiMH cells, which results in a nominal voltage of 4.8 V (1.2 V per cell). You can also use alkaline cells, which would nominally give you 6 V.

The negative battery voltage is connected to GND. The positive battery voltage is designated **VBAT**. VBAT feeds into a reverse protection circuit and then a power switching circuit controlled by the on-board pushbutton. The output of the power switching circuit is designated **VSW**.

VSW provides power to the on-board motor voltage regulator, and that regulator’s output (VM) powers the DRV8838 motor drivers, so the motors can only operate if the batteries are installed and the power switch circuit is on.

The reverse protected and switched battery voltage on VSW can be monitored through a voltage divider that is connected to **analog pin 1** (PF6). The divider outputs a voltage that is equal to one third of the battery voltage, which will be safely below the ATmega32U4’s maximum analog input voltage of 5 V. The `readBatteryMillivolts()` function in the Pololu3piPlus32U4 library can be used to determine the battery voltage from this reading.

Power switch circuit

The 3pi+ 32U4 control board uses the patented latching circuit from the **Pololu pushbutton power switch [https://www.pololu.com/product/2808]**, which provides a solid-state power switch for your robot controlled with the on-board pushbutton. By default, this pushbutton can be used to toggle power: one push turns on power and another turns it off. Alternatively, a separate pushbutton can be connected to the **PWRA** and **PWRB** pins and used instead. Multiple pushbuttons can be wired in parallel for multiple control points, and each of the parallel pushbuttons, including the one on the board itself, will be able to turn the switch on or off. **The latching circuit performs some button debouncing, but pushbuttons with excessive bouncing (several ms) might not function well with it.**

Alternatively, to disable the pushbutton, you can cut the button jumper labeled **Btn Jmp**; this allows you to connect a slide or toggle switch to control the board’s power instead. The switch should be wired such that it connects the **SW** pin to GND when it is closed, and a set of three through-holes along the left edge of the board provide a convenient place to do so (the third hole is not connected to anything but helps accommodate **3-pin switches [https://www.pololu.com/product/1408]**).

The power switch circuit also offers several alternate pushbutton connection options that result in push-on-only or push-off-only operation, and additional inputs enable further power control options like allowing your robot to turn off its own power. These advanced control options are available through the button connection pins and four control inputs:

PIN	Description
PWRA	Connect through momentary switch to pin "PWRB" for standard push-on/push-off operation. Connect through momentary switch to ground for on-only operation.
PWRB	Connect through momentary switch to pin "PWRA" for standard push-on/push-off operation.
ON	A high pulse (> 1 V) on this pin turns on the switch circuit. *This pin only functions when pushbutton operation is enabled (i.e. the button jumper has not been cut).*
OFF	A high pulse (> 1 V) on this pin turns off the switch circuit (e.g. allowing a powered device to shut off its own power). *This pin only functions when pushbutton operation is enabled.*
CTRL	With pushbutton operation enabled, this pin directly determines the state of the switch circuit. A high pulse (> 1 V) on this pin turns on the switch; a low pulse (e.g. driving the pin low with a microcontroller output line or pushing a button connected from this pin to ground) turns the switch off. Leave this pin disconnected or floating when not trying to set the switch state. Note that this pin should not be driven high at the same time the "OFF" pin is driven high.
SW	With pushbutton operation disabled (button jumper cut), this pin controls the state of the switch circuit: driving it low turns the switch on, while letting it float turns the switch off. Connect through slide or toggle switch to ground for on/off operation. Leave this pin disconnected or floating for proper pushbutton operation. We recommend only ever driving this pin low or leaving it floating; this pin should **never** be driven high while the slide switch is in the "On" position.

Motor voltage regulator

VSW supplies power to a regulator that provides 8 V for the 3pi+'s DRV8838 motor drivers. This regulated motor voltage helps keep the performance of the motors consistent as the batteries discharge and their voltage drops. However, the condition of the batteries can still have an impact on motor performance in some situations; see **Section 5.3** for more details.

The 3pi+'s motor voltage regulator is designed to cut out at a higher voltage than its 5 V logic voltage regulator. This way, if there is a significant transient drop in battery voltage due to the motors drawing a large amount of current, the motor voltage regulator will turn off and ensure that the battery voltage does not continue to fall. (Otherwise, the battery voltage could drop low enough to disable the logic regulator and cause the ATmega32U4 to reset.)

5 V and 3.3 V regulators

VSW also supplies power to a 5 V regulator, whose output is designated **R5V**. This output is not directly user-accessible, but when available, it is the preferred source for the logic power selection circuit described below. The rest of the regulator’s achievable output current, which depends on input voltage and ambient conditions, can be used to power other devices via the selection circuit’s output (5V). Under typical conditions, roughly 0.7 A of current is available from the 5 V regulator.

The 3pi+ 32U4 control board also contains a 3.3 V LDO that draws its power from the output of the logic power selection circuit. The output of the 3.3 V regulator is designated **3V3** and is used to supply the on-board inertial sensors and level shifters.

Logic power selection

The 3pi+ 32U4 control board’s power selection circuit uses the **TPS2113A power multiplexer** **[https://www.pololu.com/product/2596]** from Texas Instruments to choose whether its 5 V supply (designated **5V**) is sourced from USB or the batteries via the 5 V regulator, enabling the control board to safely and seamlessly transition between them. The TPS2113A is configured to select regulated battery power (R5V) unless the regulator output falls below about 4.5 V. If this happens, it will select the higher of the two sources, which will typically be the USB 5 V bus voltage if the control board is connected to USB.

Consequently, when 3pi+ 32U4 is connected to a computer via USB, it will receive 5 V logic power even when the power switch circuit is off. This can be useful if you want to upload or test a program without drawing power from the batteries and without operating the motors. It is safe to have USB connected and battery power switched on at the same time.

The currently selected source is indicated by the **STAT** pin; this pin is an open-drain output that is low if the external power source is selected and high-impedance if the USB supply is selected. The current limit of the TPS2113A is set to about 1.9 A nominally. For more information about the power multiplexer, see the **TPS2113A datasheet** **[https://www.pololu.com/file/0J771/tps2113a.pdf.redirect]** (1k redirect).

The 5 V output of the selection circuit is used to supply the control board’s ATmega32U4 microcontroller, logic power for the DRV8838 motor drivers, the reflectance sensors and bump sensors, and the encoders.

Power distribution

- **VBAT** is connected to the battery contact labeled **BAT+** and provides a direct connection to the battery supply.
- **VSW** is the battery voltage after reverse-voltage protection and the power switch circuit.
- **VM** is the output of the on-board 8 V motor voltage regulator.
- **R5V** is the output of the on-board 5 V logic voltage regulator. (This output is not user-

accessible.)

- **5V** is the output of the TPS2113A power multiplexer circuit which is connected to R5V by default, but switches to 5 V USB power if R5V is too low.
- **3V3** is the output of the 3.3 V LDO regulator.

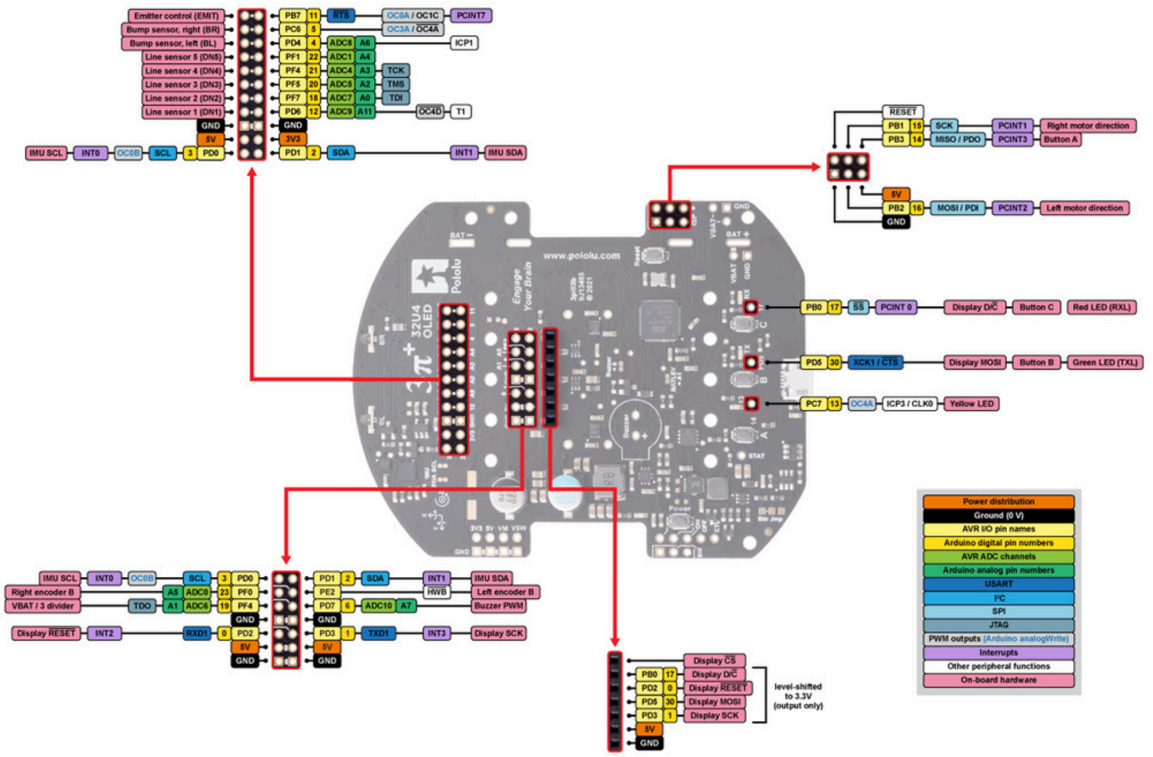
See **Section 5.8** for a diagram of the board’s power access points.

5.8. Expansion headers

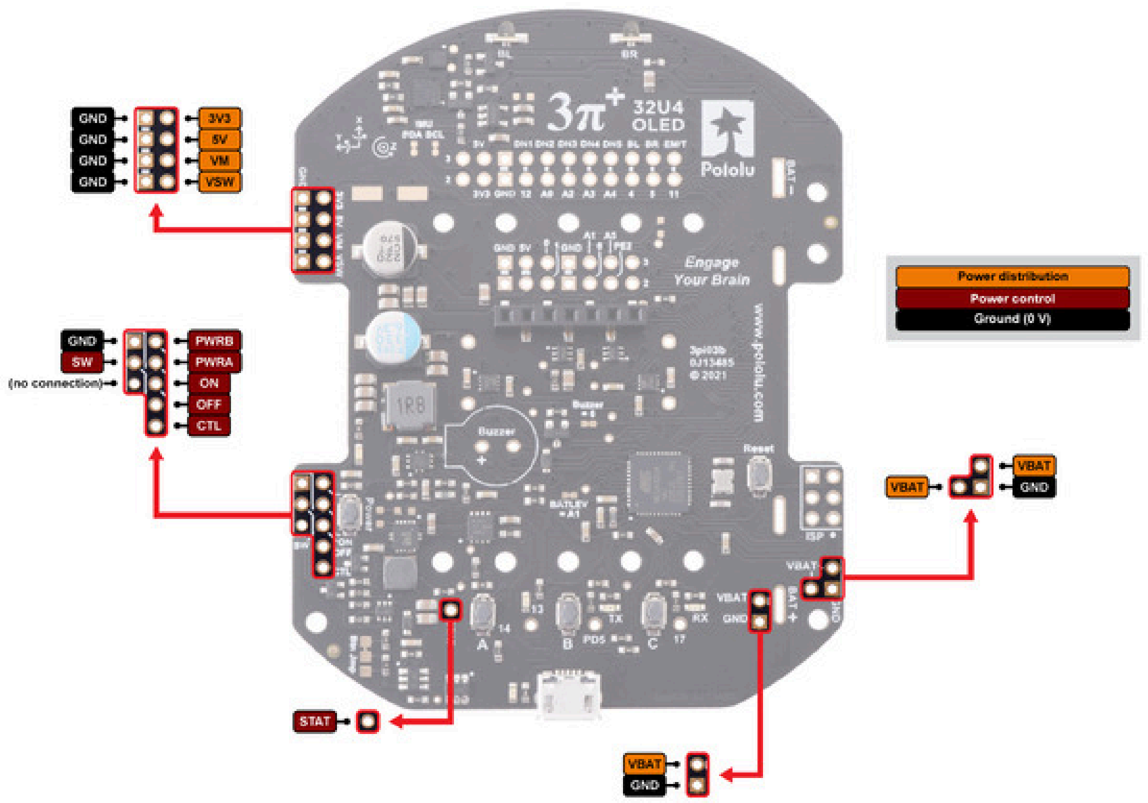
The 3pi+ 32U4 control board has several expansion headers (primarily in two areas toward the front of the board) that break out many of the general-purpose I/O lines from the ATmega32U4 microcontroller. Various power inputs, outputs, and control pins are also accessible elsewhere on the board. The following diagrams identify the locations of these pins and the hardware associated with them (OLED version pictured below). These diagrams are also available as printable PDFs:

- **3pi+ 32U4 OLED Control Board pinout and power distribution diagrams [https://www.pololu.com/file/0J1823/3pi-plus-32u4-oled-control-board-pinout.pdf]** (1MB pdf)
- **3pi+ 32U4 Control Board pinout and power distribution diagrams [https://www.pololu.com/file/0J1791/3pi-plus-32u4-control-board-pinout.pdf]** (1MB pdf) *(original LCD version)*

For more information about the ATmega32U4 microcontroller and its peripherals, see Atmel’s ATmega32U4 documentation.



3pi+ 32U4 OLED Control Board pinout and peripherals.



3pi+ 32U4 OLED Control Board power distribution and control.

5.9. Pin assignments

The table below lists the most important pin assignments for the ATmega32U4 on the 3pi+ 32U4. This table is helpful if you want to add your own electronics to the 3pi+ 32U4, write your own low-level code for interfacing with the hardware, or just want to understand better how the 3pi+ 32U4 works. Each row represents a physical pin on the ATmega32U4.

The “ATmega32U4 pin name” column shows the official name of the pin according to the **ATmega32U4 datasheet** [https://www.microchip.com/wwwproducts/en/ATmega32u4].

The “Arduino pin names” column lists the names provided by the Arduino environment for the pin. These names can generally be used as arguments to any function that takes a pin number. However, there are some exceptions. For example, passing the number 4 to `analogRead` actually reads pin A4, not pin 4. Also, due to hardware limitations, some functions only work on a limited set of pins.

The “3pi+ 32U4 functions” column documents what the pin is used for on the 3pi+ 32U4. Many pins can serve multiple purposes concurrently by switching modes. For example, PB0 can read the state of button C when it is an input, and it can control the red LED and serve as a display interface line when it is an output.

The “Note/alternate functions” column documents other features of the pin, although some of those features might be impractical to use.

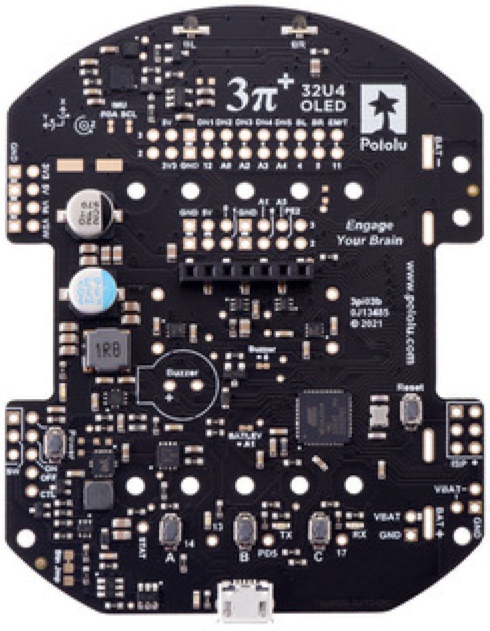
ATmega32U4 pin name	Arduino pin names	3pi+ 32U4 functions	Notes/alternate functions
PD6	12, A11, 29	Line sensor 1 (DN1, leftmost)	Analog input (ADC7) JTAG test data in (TDI)
PF7	A0, 18	Line sensor 2 (DN2)	Analog input (ADC5) JTAG test mode select (TMS)
PF5	A2, 20	Line sensor 3 (DN3, center)	Analog input (ADC4) JTAG test clock (TCK)
PF4	A3, 21	Line sensor 4 (DN4)	Analog input (ADC8) Timer1 input capture pin (ICP1)
PF1	A4, 22	Line sensor 5 (DN5, rightmost)	Analog input (ADC9) Timer4 PWM output D ($\overline{\text{OC4D}}$) Timer1 counter source (T1)
PD4	4, A6, 24	Left bump sensor (BL)	Analog input (ADC1)
PC6	5	Right bump sensor (BR)	Timer3 PWM output A (OC3A) Timer4 PWM output A ($\overline{\text{OC4A}}$)
PB7	11	IR emitter control (EMIT) (for line sensors and bump sensors)	Timer0 PWM output A (OC0A) Timer1 PWM output C (OC1C) UART flow control ($\overline{\text{RTS}}$) Pin-change interrupt (PCINT7)
PD2	0	*OLED version:* Display reset ($\overline{\text{RESET}}$) *LCD version:* LCD control line (RS)	UART receive pin (RXD1) External interrupt source ($\overline{\text{INT2}}$)
PD3	1	*OLED version:* Display SPI clock (SCK) *LCD version:* LCD control line (E)	UART transmit pin (TXD1) External interrupt source ($\overline{\text{INT3}}$)
PB3	14, MISO	User pushbutton A *LCD version:* LCD data line DB4	SPI Master Input/Slave Output (MISO) Pin-change interrupt (PCINT3)
PB0	17, LED_BUILTIN_RX, SS	Red LED (RX) User pushbutton C *OLED version:* Display data/command select (D/$\overline{\text{C}}$) *LCD version:* LCD data line DB5	SPI slave select ($\overline{\text{SS}}$) Pin-change interrupt (PCINT0)
PC7	13, LED_BUILTIN	Yellow LED *LCD version:* LCD data line DB6	Timer4 PWM output A (OC4A) Timer3 input capture pin (ICP3) Divided system clock output

ATmega32U4 pin name	Arduino pin names	3pi+ 32U4 functions	Notes/alternate functions
			(CLKO)
PD5	30, LED_BUILTIN_TX	Green LED (TX) User pushbutton B *OLED version:* Display data (MOSI) *LCD version:* LCD data line DB7	UART external clock (XCK1) UART flow control ($\overline{\text{CTS}}$)
PD7	6, A7, 25	Buzzer PWM	Analog input (ADC10) Timer4 PWM output D (OC4D) Timer0 counter source (T0)
PF6	A1, 19	Battery level input (VSW/3)	Analog input (ADC6) JTAG test data out (TDO)
PB6	10, A10, 28	Left motor PWM	Analog input (ADC13) Timer1 PWM output B (OC1B) Timer4 PWM output B (OC4B) Pin-change interrupt (PCINT6)
PB2	16, MOSI	Left motor direction	SPI Master Output/Slave Input (MOSI) Pin-change interrupt (PCINT2)
PB5	9, A9, 27	Right motor PWM	Analog input (ADC12) Timer1 PWM output A (OC1A) Timer4 PWM output B ($\overline{\text{OC4B}}$) Pin-change interrupt (PCINT5)
PB1	15, SCK	Right motor direction	SPI Clock (SCK) Pin-change interrupt (PCINT1)
PB4	8, A8, 26	Left encoder XORed input	Analog input (ADC11) Pin-change interrupt (PCINT4)
PE2	-	Left encoder input	Hardware bootloader select ($\overline{\text{HWB}}$)
PE6	7	Right encoder XORed input	Analog comparator negative input (AIN0) External interrupt source (INT6)
PF0	A5, 23	Right encoder input	Analog input (ADC0)
PD0	3, SCL	I²C clock for inertial sensors	Timer0 PWM output B (OC0B)

ATmega32U4 pin name	Arduino pin names	3pi+ 32U4 functions	Notes/alternate functions
			External interrupt source ($\overline{\text{INT0}}$)
PD1	2, SDA	I²C data for inertial sensors	External interrupt source ($\overline{\text{INT1}}$)
$\overline{\text{RESET}}$	-	Reset pushbutton	internally pulled high, active low
AREF	-	-	Analog reference

5.10. Adding electronics

This section gives tips for how the 3pi+ 32U4 can be expanded with additional electronics.



Freeing up I/O pins

If you want your additional electronics to send or receive information from the AVR, you will need to connect them to one or more of the AVR's I/O pins. Each I/O pin is already being used for some other purpose, as documented in **Section 5.9**, so you might need to disable or disconnect one of the other features of the 3pi+ 32U4.

If you do not need some or all of the reflectance sensors or bump sensors, you can free up as many as 8 pins for other purposes: pin 12 (PD6), pin 18 (A0/PF7), pin 20 (A2/PF5), pin 21 (A3/PF4), pin 22 (A4/PF1), pin 4 (PD4), pin 5 (PC6), and pin 11 (PB7). Each pin can be used for digital input and output, while 5 of them (all except pin 5 and pin 11) can be used as analog inputs. Pin 5 can also be used for PWM output. If you want to use these pins as digital or analog inputs, you might need to disconnect them from the sensors by cutting the trace between the appropriate pair of through-holes on the underside of the board (see **Section 5.5**). If you only want to use one of these pins as an output, you might not need to cut its trace.

If you do not need the AVR to be able to measure the battery voltage, you can use pin 19 (A1, PF6) for other purposes. This pin can be used for digital input and output, as well as analog input. If you want to use this pin as a digital or analog input, you might need to cut the surface-mount jumper labeled "BATLEV = A1" in order to disconnect it from the VBAT voltage divider. If you only want to use A1 as an output, you might not need to cut that jumper.

If you do not need the display, you can remove it. This frees up pin 0 (PD2) and pin 1 (PD3). These pins are the transmit (TX) and receive (RX) pins of the UART, so you can use them to establish serial communication with another microcontroller. These pins are also capable of digital I/O. These pins are the recommended pins for connecting two output channels from an RC receiver, or for controlling two RC servos, because they are arranged in a convenient way with respect to power and ground on the right-side expansion header.

On the original (LCD) version of the 3pi+ 32U4, if you have removed the LCD and do not need to use button A, this frees up pin 14 (PB3). Pin 14 is capable of digital input and output. Removing the LCD also frees up the LCD contrast potentiometer for other purposes. The output of the potentiometer is a 0 V to 5 V signal which is accessible on the LCD connector. It can be connected to any free analog input if you want to read it from the AVR, or it might be useful to connect it to the other electronics that you are adding.

If you do not need to use the buzzer, you can cut the surface-mount jumper labeled “Buzzer = 6”. This disconnects pin 6 (PD7) from the buzzer, so it can be used for other things. Pin 6 (PD7) can be used as a PWM output, digital I/O line, or analog input. Disabling the buzzer also frees up Timer4, which has several PWM output pins. These pins can be used as PWM outputs if they are not needed for their normal tasks.

Be careful about connecting electronics to pin 13 (PC7), pin 17 (PB0), and PD5. These pins are used to control the LEDs on the 3pi+ 32U4. All three of these pins are controlled as outputs by the bootloader. Pin 17 (PB0) and PD5 are used as RX and TX indicators, so if you are sending or receiving data over USB then the Arduino USB code will drive those pins in its interrupt service routines while your sketch is running.

It should be possible to attach additional I²C slave devices to the 3pi+ 32U4’s I²C bus without giving up any features as long as the additional devices’ slave addresses do not conflict with those of the inertial sensors. (The sensors’ addresses are specified in their respective datasheets, which can be found in **Section 5.6**). The I²C pins (pins 2 and 3) operate at 5 V, so level shifters might be necessary to interface with other devices that use different voltages. (The level-shifted 3.3 V signals used by the inertial sensors are not available to the user.)

If you do not want to use the inertial sensors on the 3pi+ 32U4’s I²C bus, you can cut the surface-mount jumpers labeled “SDA” and “SCL” under “IMU”. This frees up pin 2 (PD1) and pin 3 (PD0). These pins can be used as digital inputs and outputs.

Power

Many of the 3pi+’s power nodes are accessible from the header at the front left of the board. If you power additional devices from VSW, then they will be powered whenever the 3pi+’s power switch is in the ON position, and they will receive whatever voltage the batteries are outputting. If you power them from VM, they will get 8 V power, shared with the motor drivers, whenever the batteries are installed and the power switch circuit is on (but they cannot be powered from USB). If you power them from the 5V pin, then they will receive 5V power whenever the 3pi+ 32U4 logic components are powered. If you power them from 3V3, they will receive 3.3V power whenever the 3pi+ 32U4 logic components are powered. For more information about these power nodes and how much current they can provide, see **Section 5.7**.

It is also possible to add your own power switch to control power to the 3pi+, as described in **Section 5.7**.

Ground

You should make sure that all the grounds in your system are connected. The 3pi+ 32U4's ground node can be accessed from pins labeled "GND". It should be connected to the ground node of every other circuit board or device you add to the robot.

5.11. AVR timers

The ATmega32U4 has 4 timers: Timer0, Timer1, Timer3, and Timer4. Each timer has a different set of features, as documented in the datasheet.

- Timer0 is used by the Arduino environment for timing-related functions like `millis()` .
- Timer1 is used by the Pololu3piPlus32U4 Arduino library for driving motors.
- Timer3 is *not* used by the Pololu3piPlus32U4 Arduino library and can be freely used for your own purposes.
- Timer4 is used by the Pololu3piPlus32U4 Arduino library for controlling the buzzer.

5.12. Schematics and dimensions

Schematics

The schematic diagram for the 3pi+ 32U4 Control Board is available as a PDF:

- **3pi+ 32U4 OLED Control Board schematic diagram [https://www.pololu.com/file/0J1822/3pi-plus-32u4-oled-control-board-schematic.pdf]** (391k pdf)
- **3pi+ 32U4 Control Board schematic diagram [https://www.pololu.com/file/0J1790/3pi-plus-32u4-control-board-schematic.pdf]** (584k pdf) *(original LCD version)*

Dimensions

Basic dimension diagrams are available as PDFs for the 3pi+ 32U4 Control Board by itself as well as the assembled 3pi+ 32U4 robot:

- **3pi+ 32U4 OLED Control Board dimension diagram [https://www.pololu.com/file/0J1825/3pi-**

plus-32u4-oled-control-board-dimensions.pdf] (1MB pdf)

- **3pi+ 32U4 OLED Robot dimension diagram [https://www.pololu.com/file/0J1827/3pi-plus-32u4-oled-robot-dimensions.pdf]** (1MB pdf)
- **3pi+ 32U4 Control Board dimension diagram [https://www.pololu.com/file/0J1792/3pi-plus-32u4-control-board-dimensions.pdf]** (1MB pdf) *(original LCD version)*
- **3pi+ 32U4 Robot dimension diagram [https://www.pololu.com/file/0J1796/3pi-plus-32u4-robot-dimensions.pdf]** (1MB pdf) *(original LCD version)*

Dimensions that are not included in the above diagrams can be measured from the following DXF drawings:

- **3pi+ 32U4 OLED Control Board drill guide [https://www.pololu.com/file/0J1824/3pi03b-drill.dxf]** (216k dxf)
- **3pi+ 32U4 OLED Robot front, top, and side views [https://www.pololu.com/file/0J1830/3pi-plus-32u4-oled-robot-dxf.zip]** (453k zip)
- **3pi+ 32U4 Control Board drill guide [https://www.pololu.com/file/0J1789/3pi03a-drill.dxf]** (216k dxf) *(original LCD version)*
- **3pi+ 32U4 Robot front, top, and side views [https://www.pololu.com/file/0J1797/3pi-plus-32u4-robot-dxf.zip]** (471k zip) *(original LCD version)*

3D models of the 3pi+ 32U4 Control Board and robot are also available in STEP format:

- **3pi+ 32U4 OLED Control Board 3D model [https://www.pololu.com/file/0J1826/3pi-plus-32u4-oled-control-board.step]** (30MB step)
- **3pi+ 32U4 OLED Robot 3D models [https://www.pololu.com/file/0J1828/3pi-plus-32u4-oled-robot-step.zip]** (19MB zip)
  - Note: This model uses simplified models of the control electronics to reduce the file size.
- **3pi+ 32U4 Control Board 3D model [https://www.pololu.com/file/0J1788/3pi-plus-32u4-control-board.step]** (28MB step) *(original LCD version)*
- **3pi+ 32U4 Robot 3D models [https://www.pololu.com/file/0J1793/3pi-plus-32u4-robot-step.zip]** (18MB zip) *(original LCD version)*
  - Note: This model uses simplified models of the control electronics to reduce the file size.

6. Programming the 3pi+ 32U4

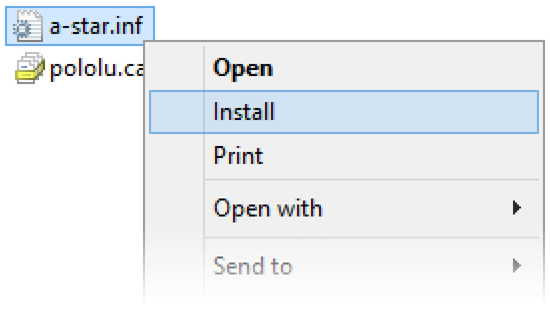
The 3pi+ 32U4 is designed to be programmed over USB from the **Arduino IDE** **[http://arduino.cc/en/Main/Software]**. It can be programmed from Windows, Linux, and macOS. The ATmega32U4 on the control board comes preloaded with the same USB bootloader as the **A-Star 32U4 family** **[https://www.pololu.com/category/149/a-star-programmable-controllers]** of general-purpose programmable ATmega32U4 boards. The following sections will help you get started programming your 3pi+ 32U4.

6.1. Installing Windows drivers

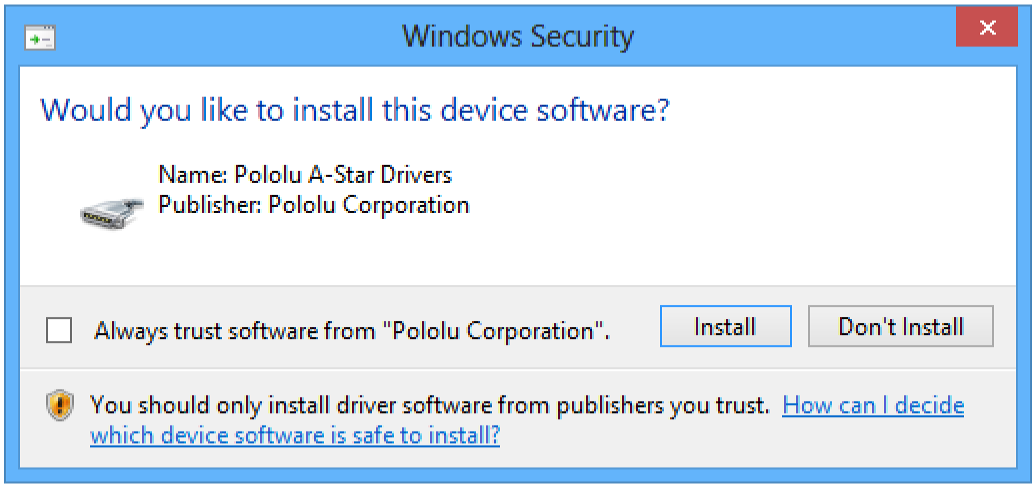
If you use Windows XP, you will need to have either Service Pack 3 or Hotfix KB918365 installed before installing the A-Star drivers. Some users who installed the hotfix have reported problems that were solved by upgrading to Service Pack 3, so we recommend Service Pack 3 over the hotfix.

Before you connect your Pololu A-Star 32U4 (or another of our 32U4 family of boards) to a computer running Microsoft Windows, you should install its drivers:

1. Download the **A-Star Windows Drivers** **[https://www.pololu.com/file/0J1240/a-star-windows-1.3.0.0.zip]** (7k zip) and extract the ZIP file to a temporary folder on your computer. (These files are also available in the "drivers" directory from the **A-Star repository on GitHub** **[https://github.com/pololu/a-star]**.)
2. Open the "a-star-windows" folder. Right-click on "a-star.inf" and select "Install".



3. Windows will ask you whether you want to install the drivers. Click "Install" (Windows 10, 8, 7, and Vista) or "Continue Anyway" (Windows XP).



Would you like to install this device software?

Name: Pololu A-Star Drivers
Publisher: Pololu Corporation

Always trust software from "Pololu Corporation".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

4. Windows will not tell you when the installation is complete, but it should be done after a few seconds.

Windows 10, Windows 8, Windows 7, and Windows Vista users: After installing the drivers, your computer should automatically recognize the device when you connect it via USB. No further action from you is required. However, the first time you connect an A-Star device to your computer, Windows will take several seconds to recognize the device and configure itself properly. The first time you program the device, Windows will again take several seconds to recognize the A-Star USB bootloader, and this could cause the programming operation to fail the first time. Also, Windows will need to re-recognize the device and the bootloader if you connect the board to another USB port that it has not been connected to before.

Windows XP users: After installing the drivers, you will need to follow steps 5–9 for each new A-Star device you connect to your computer. You will also need to follow these steps the first time you attempt to program the device in order to make Windows recognize the bootloader, and when you connect the device to a different USB port that it has not been connected to before.

5. Connect the device to your computer's USB port.
6. When the "Found New Hardware Wizard" is displayed, select "No, not this time" and click "Next".
7. On the second screen of the "Found New Hardware Wizard", select "Install the software automatically" and click "Next".
8. Windows XP will warn you again that the driver has not been tested by Microsoft and recommend that you stop the installation. Click "Continue Anyway".
9. When you have finished the "Found New Hardware Wizard", click "Finish".

COM port details

After installing the drivers and plugging in an A-Star, in the "Ports (COM & LPT)" category of the Device Manager, you should see a COM port for the A-Star's running sketch named "Pololu A-Star 32U4".

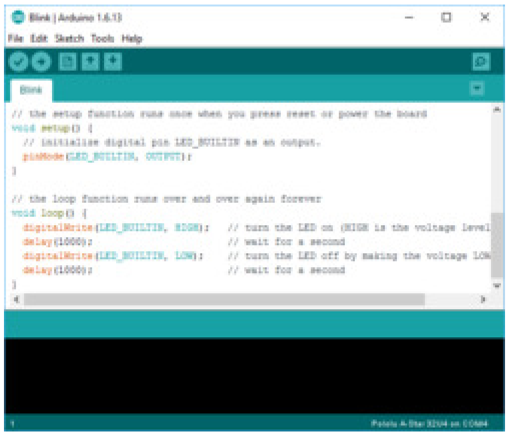
You might see that the COM port is named "USB Serial Device" in the Device Manager instead of having a descriptive name. This can happen if you are using Windows 10 or later and you plugged the A-Star into your computer before installing our drivers for it. In that case, Windows will set up your A-Star using the default Windows serial driver (usbser.inf), and it will display "USB Serial Device" as the name for the port. The port will still be usable, but it will be hard to tell if it is the right one because of the generic name shown in the Device Manager. We recommend fixing the names in the Device Manager by right-clicking on each "USB Serial Device" entry, selecting "Update Driver Software…", and then selecting "Search automatically for updated driver software". Windows should find the drivers you already installed, which contain the correct name for the port.

If you are using Windows 10 or later and choose not to install the drivers, the A-Star will still be usable. To tell which "USB Serial Device" in your Device Manager is the A-Star, double-click on each one and look at the "Hardware Ids" property in the "Details" tab. An A-Star running a sketch will have the ID `USB\VID_1FFB&PID_2300&MI_00` , while an A-Star in bootloader mode will have the ID `USB\VID_1FFB&PID_0101` .

If you want to change the COM port numbers assigned to your A-Star, you can do so using the Device Manager. Double-click a COM port to open its properties dialog, and click the "Advanced…" button in the "Port Settings" tab.

6.2. Programming using the Arduino IDE

Our 32U4 family of boards can be programmed from the popular Arduino integrated development environment (IDE). The Arduino IDE is a cross-platform, open source application that integrates a C++ code editor, the GNU C++ compiler, and a program upload utility. To get started programming your device with the Arduino IDE (version 1.6.4 or later), follow these steps:



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

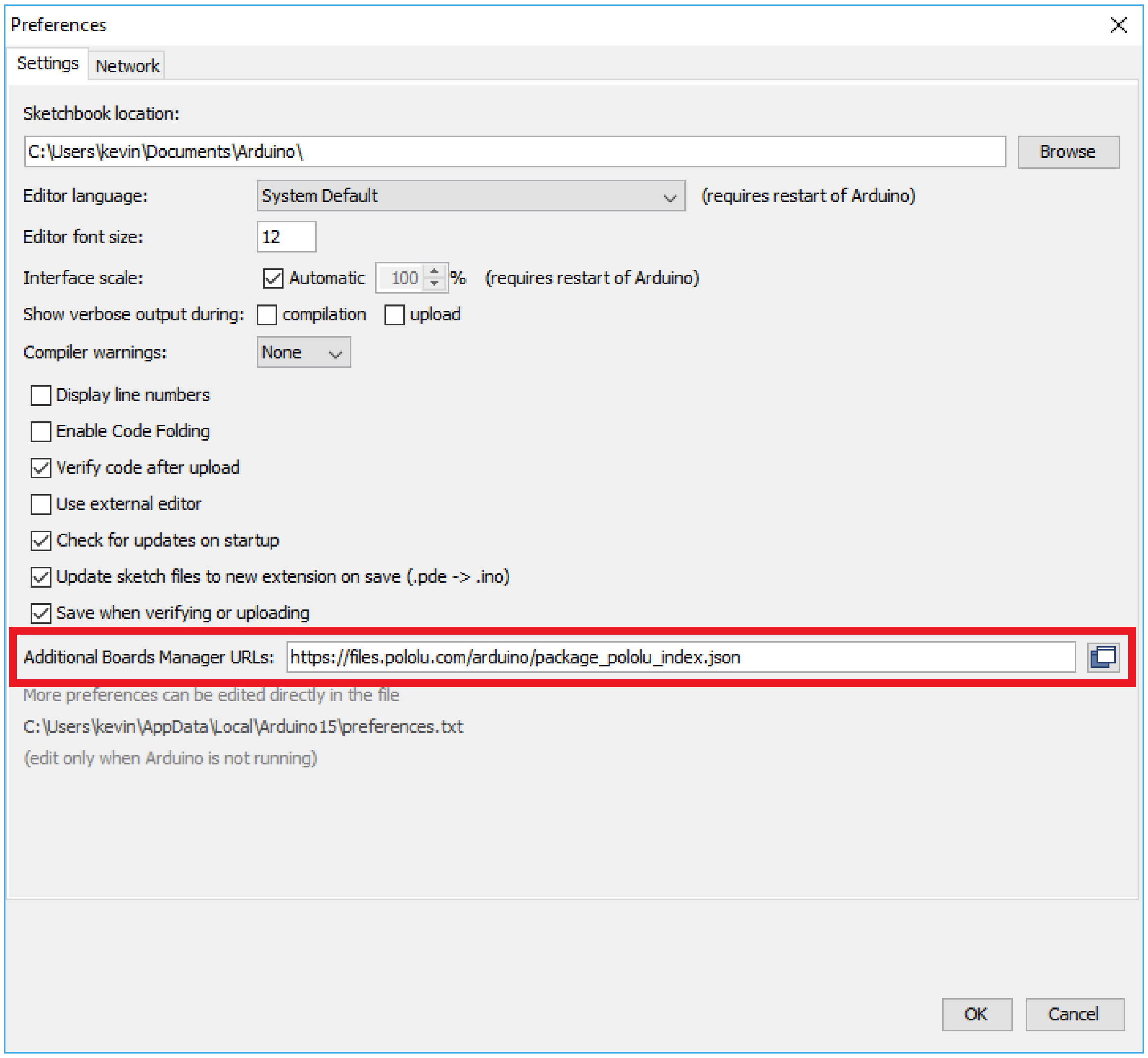
Programming the A-Star 32U4 from the Arduino IDE.

1. Download the Arduino IDE from the **Arduino Download page [http://arduino.cc/en/Main/Software]**, install it, and start it.
2. In the Arduino IDE, open the **File** menu (Windows/Linux) or the **Arduino** menu (macOS) and select "Preferences".

3. In the Preferences dialog, find the "Additional Boards Manager URLs" text box (highlighted in the picture below). Copy and paste the following URL into this box:

   https://files.pololu.com/arduino/package_pololu_index.json

   If there are already other URLs in the box, you can either add this one separated by a comma **or** click the button next to the box to open an input dialog where you can add the URL on a new line.



Sketchbook location: C:\Users\kevin\Documents\Arduino\

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

[ ] Display line numbers
[ ] Enable Code Folding
[✓] Verify code after upload
[ ] Use external editor
[✓] Check for updates on startup
[✓] Update sketch files to new extension on save (.pde -> .ino)
[✓] Save when verifying or uploading

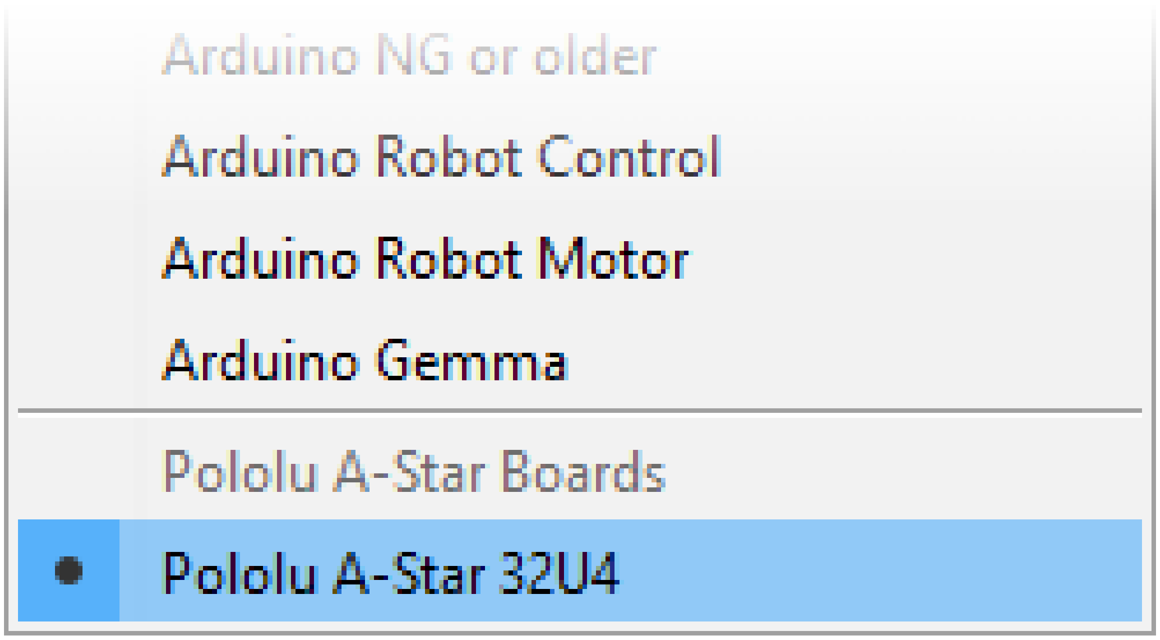
Additional Boards Manager URLs: https://files.pololu.com/arduino/package_pololu_index.json

More preferences can be edited directly in the file
C:\Users\kevin\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Adding a Boards Manager index for Pololu boards in the Arduino IDE's Preferences dialog.

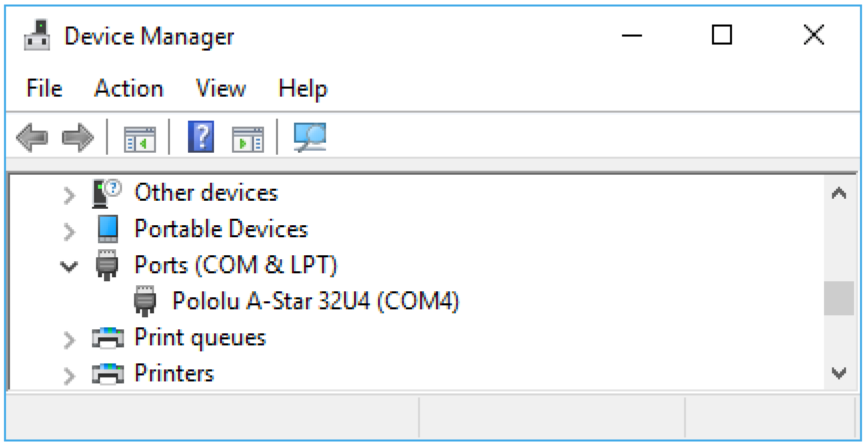
4. Click the "OK" button to close the Preferences dialog.

5. In the **Tools > Board** menu, select "Boards Manager…" (at the top of the menu).
6. In the Boards Manager dialog, search for "Pololu A-Star Boards".
7. Select the "Pololu A-Star Boards" entry in the list, and click the "Install" button.
8. After the installation finishes, click the "Close" button to close the Boards Manager dialog.
9. In the **Tools > Board** menu, select the "Pololu A-Star 32U4" entry. If you do not see your device listed in the Board menu, try restarting the Arduino IDE.



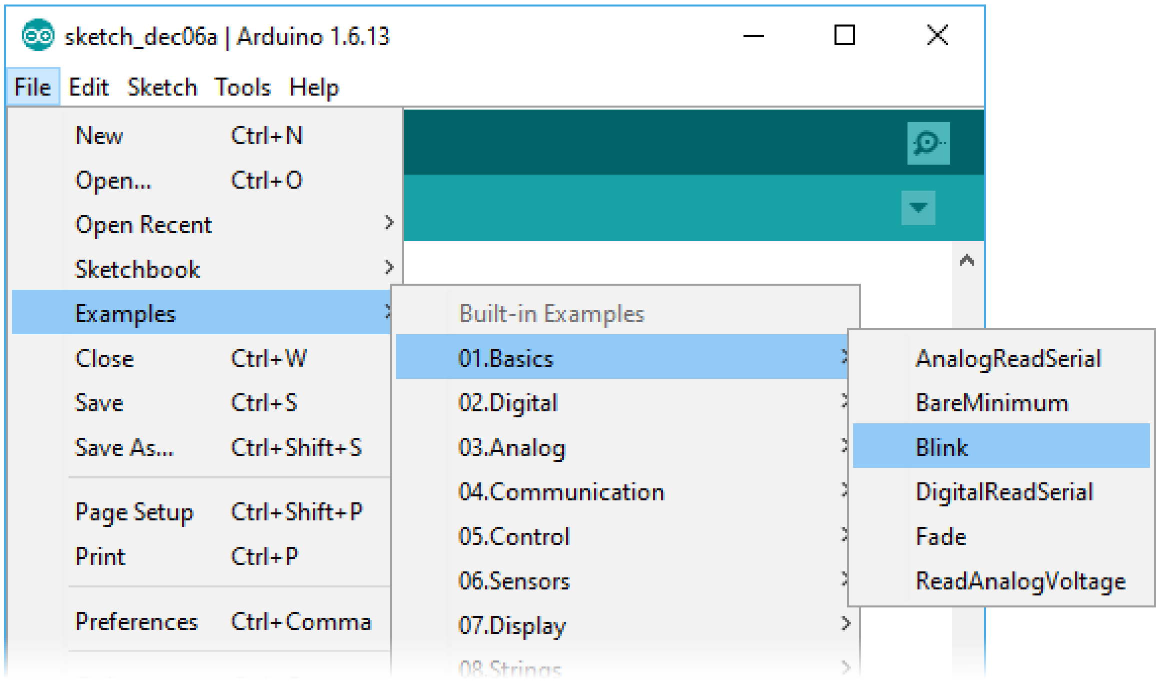
Selecting the Pololu A-Star 32U4 in the Boards menu.

10. In the **Tools > Port** menu, select the port for the device. On Windows you can determine what COM port the device is assigned to by looking at the "Ports (COM & LPT)" section of the Device Manager. On Linux, the port name will begin with "/dev/ttyACM". On Mac OS X, the port name will begin with "/dev/tty.usbmodem".



Windows 10 Device Manager showing the A-Star's virtual COM port.

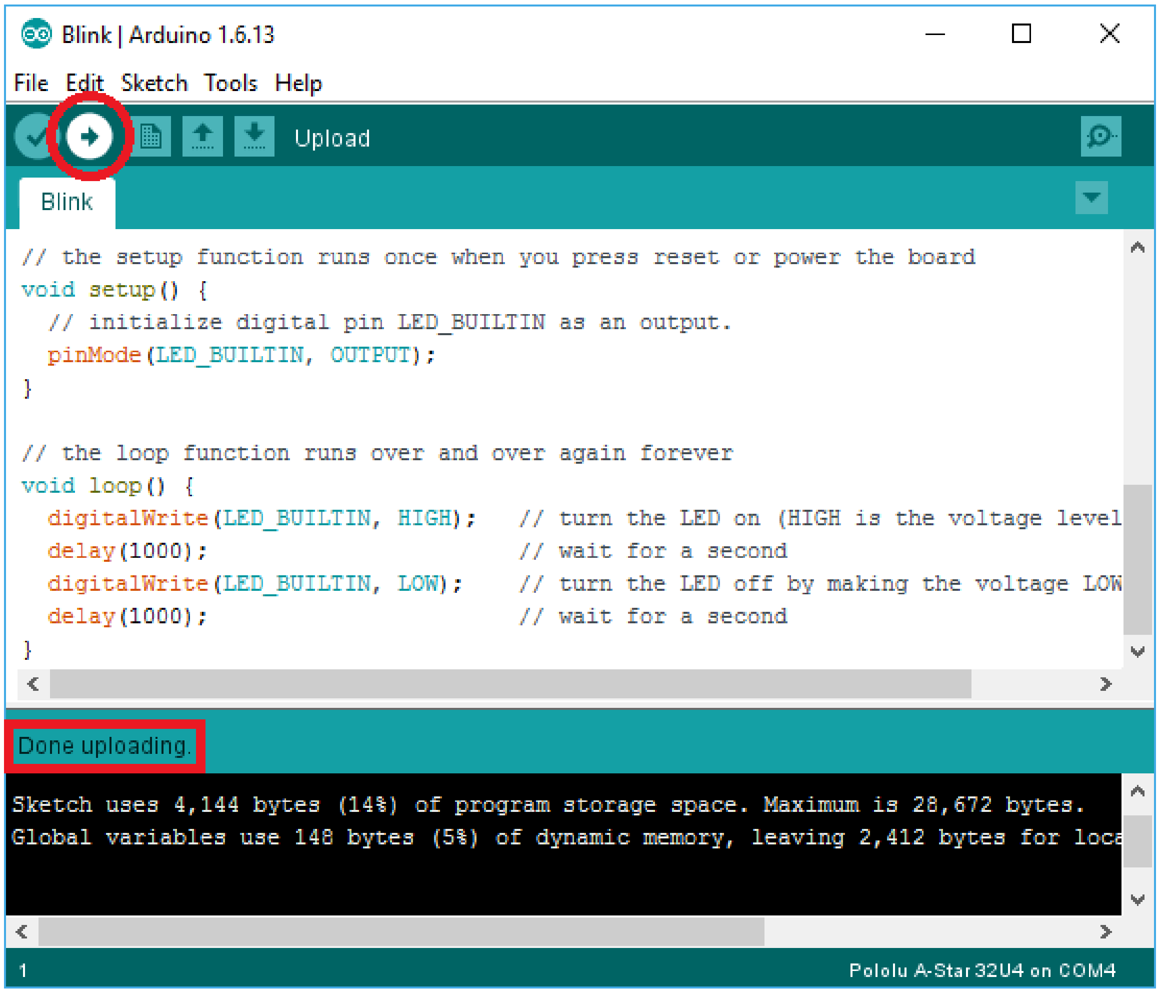
11. Open up the "Blink" Arduino example, which can be found under **File > Examples > 01.Basics > Blink**. The code in this example will blink the yellow LED. When you select the Blink example, a new Arduino IDE window will open up. It is OK to close the first window.



Selecting the Blink example in the Arduino IDE.

12. Press the "Upload" button to compile the sketch and upload it to the device. If everything goes correctly, you will see the message "Done uploading" appear near the bottom of the window. If you are using Windows and you have not previously programmed an A-Star device on this

USB port, then Windows might take several seconds to recognize the A-Star bootloader. The bootloader times out after 8 seconds and goes back to running the sketch, so the upload might fail if Windows does not recognize it quickly enough. If this happens, try again. If you are using Windows XP and have not programmed an A-Star on this USB port, you will have to go through the Found New Hardware Wizard again as described in the previous section, but the second time you try to upload it should work. If the Arduino IDE has trouble connecting to the port or using it, try unplugging the device, closing any programs that might be using the serial port, restarting the Arduino IDE, and then plugging the device back in.



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Done uploading.

```
Sketch uses 4,144 bytes (14%) of program storage space. Maximum is 28,672 bytes.
Global variables use 148 bytes (5%) of dynamic memory, leaving 2,412 bytes for loca
```

Uploading a sketch to the A-Star using the Arduino IDE.

13. If you uploaded the Blink sketch, then the yellow LED should be blinking once every two seconds. However, we ship some A-Stars with that same example already programmed onto it, so you might not be convinced that anything has changed. Try changing the delay values in the sketch to something else and uploading again to see if you can change the speed of

the LED.

The A-Star 32U4 boards are similar enough to the Arduino Leonardo that you do not actually have to install the add-on. If you want to, you can just select the "Arduino Leonardo" board in the Arduino IDE. Note that if you upload a sketch to the device this way, your computer will then recognize it as a Leonardo (for example, its entry in the Windows Device Manager will display "Arduino Leonardo").

After you succeed in programming your device from the Arduino IDE, there are many resources you can use to learn more:

- The Arduino IDE has many **examples** **[http://arduino.cc/en/Tutorial/HomePage]** that can run on A-Stars.
- The Arduino website has a **Language Reference** **[http://arduino.cc/en/Reference/HomePage]**, a wiki called the **The Arduino Playground** **[http://playground.arduino.cc/]**, and other resources.
- The A-Star 32U4 boards are similar to the **Arduino Leonardo** **[https://www.pololu.com/product/2192]** and **Arduino Micro** **[https://www.pololu.com/product/2188]**, so you can search the Internet for relevant projects that use one of those boards.
- The Related Resources section lists many more resources.

Disabling ModemManager in Linux

If you are using Linux and have trouble following the instructions above, your issue might be caused by a program called ModemManager. This program automatically connects to serial ports and sends modem commands to them, interfering with other software using those ports. You can run `ps ax | grep -i Modem` to see if ModemManager is running. On Ubuntu, the command to permanently disable ModemManager is:

```
sudo systemctl disable ModemManager
```

6.3. Programming using avr-gcc and AVRDUDE

This section explains how to program our 32U4 family of boards using the avr-gcc toolchain and AVRDUDE. This section is intended for advanced users who do not want to use the Arduino IDE as described in the previous section.

Getting the prerequisites

If you are using Windows, we recommend downloading **WinAVR** **[http://winavr.sourceforge.net/]**, which contains the avr-gcc toolchain and a command-line utility called **AVRDUDE** **[http://www.nongnu.org/**

avrdude/] that can be used to upload programs to the A-Star bootloader. If the version of GNU Make that comes with WinAVR crashes on your computer, we recommend using the **Pololu version of GNU Make [https://github.com/pololu/make/releases]**.

If you are using macOS, we recommend first installing **Homebrew [https://brew.sh/]**. Then run the following commands to install AVRDUDE and **homebrew-avr [https://github.com/osx-cross/homebrew-avr]**:

```
brew install avrdude
xcode-select --install
brew tap osx-cross/avr
brew install avr-gcc
```

If you are using Linux, you will need to install avr-gcc, avr-libc, and AVRDUDE. Ubuntu users can get the required software by running:

```
sudo apt-get install gcc-avr avr-libc avrdude
```

After you have installed the prerequisites, open a command prompt and try running these commands to make sure all the required utilities are available:

```
avr-gcc -v
avr-objcopy -V
make -v
avrdude
```

If any of those commands fail, make sure the desired executable is installed on your computer and make sure that it is in a directory listed in your PATH environment variable.

Compiling an example program

Copy the following code to a file named "main.c":

```
#define F_CPU 16000000
#include <avr/io.h>
#include <util/delay.h>

int main()
{
  DDRC |= (1 << DDC7);    // Make pin 13 be an output.
  while(1)
  {
    PORTC |= (1 << PORTC7);   // Turn the LED on.
    _delay_ms(500);
    PORTC &= ~(1 << PORTC7);  // Turn the LED off.
    _delay_ms(500);
  }
}
```

In the same folder, create a file named "Makefile" with the following contents:

```
PORT=\\\\.\\GLOBALROOT\\Device\\USBSER000
```

```
MCU=atmega32u4
CFLAGS=-g -Wall -mcall-prologues -mmcu=$(MCU) -Os
LDFLAGS=-Wl,-gc-sections -Wl,-relax
CC=avr-gcc
TARGET=main
OBJECT_FILES=main.o

all: $(TARGET).hex

clean:
	rm -f *.o *.hex *.obj *.hex

%.hex: %.obj
	avr-objcopy -R .eeprom -O ihex $< $@

%.obj: $(OBJECT_FILES)
	$(CC) $(CFLAGS) $(OBJECT_FILES) $(LDFLAGS) -o $@

program: $(TARGET).hex
	avrdude -p $(MCU) -c avr109 -P $(PORT) -U flash:w:$(TARGET).hex
```

Make sure that the PORT variable in the Makefile is the name of the device’s virtual serial port. In Windows, `\\\\.\\GLOBALROOT\\Device\\USBSER000` should work if the A-Star is the only USB device connected that is using the usbser.sys driver, but you can change it to be the actual name of the COM port (e.g. `COM13`).

In a command prompt, navigate to the directory with the Makefile and main.c. If you run the command `make` , the code should get compiled and produce a file named “main.hex”.

Programming

To program the A-Star device, you will need to get it into bootloader mode first. One way to do this is to reset the AVR twice within 750 ms. Most of the boards in our 32U4 family have a reset button that can be used to reset the board. On any of our 32U4 family of boards, a pushbutton can be connected between the GND and RST pins to serve as a reset button, or you can use a wire. Once the device is in bootloader mode, quickly run the command `make program` to program it. If you wait longer than 8 seconds, the A-Star bootloader will exit and the AVR will go back to running the user program.

7. Pololu3piPlus32U4 Arduino library

The 3pi+ 32U4 can be programmed from the Arduino IDE as described in the preceding sections.

To help interface with all the on-board hardware on the control board, we provide the **Pololu3piPlus32U4 library**. The **Pololu3piPlus32U4 library documentation** **[https://pololu.github.io/pololu-3pi-plus-32u4-arduino-library]** provides detailed information about the library, and the library comes with several example sketches.

Use the Library Manager in version 1.8.10 or later of the Arduino software (IDE) to install this library:

1. In the Arduino IDE, open the “Tools” menu and select “Manage Libraries…”.
2. Search for “3pi+”.
3. Click the Pololu3piPlus32U4 entry in the list.
4. Click “Install”.
5. If you see a prompt asking to install missing dependencies, click “Install all”.

After you install the Pololu3piPlus32U4 library, you can learn more about it by trying the included example sketches.

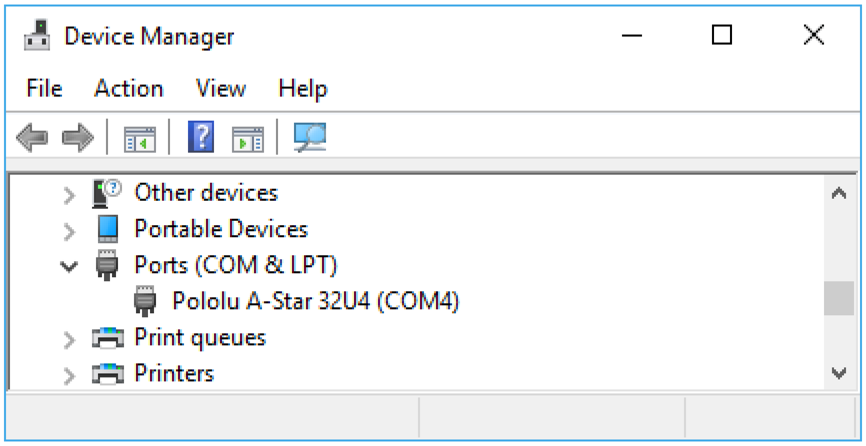
8. The 3pi+ 32U4 USB interface

Our 32U4 family of boards are based on a single ATmega32U4 AVR microcontroller that runs the user program and also handles the USB connection to the computer. The AVR has a full-speed USB transceiver built into it and can be programmed to present almost any type of USB device interface to the computer.

USB is an asymmetric system that consists of a single "host" connected to multiple "devices". The host is typically a personal computer. The ATmega32U4 can only act as a USB device, so an A-Star device cannot be connected to other USB devices like mice and keyboards; it can only be connected to a host such as your computer.

Programming an ATmega32U4 board using the Arduino IDE as described earlier will automatically configure it as a composite device with a single virtual serial port. If you program the microcontroller with an Arduino sketch that implements another USB device class, like HID or MIDI, you will see additional child devices as well.

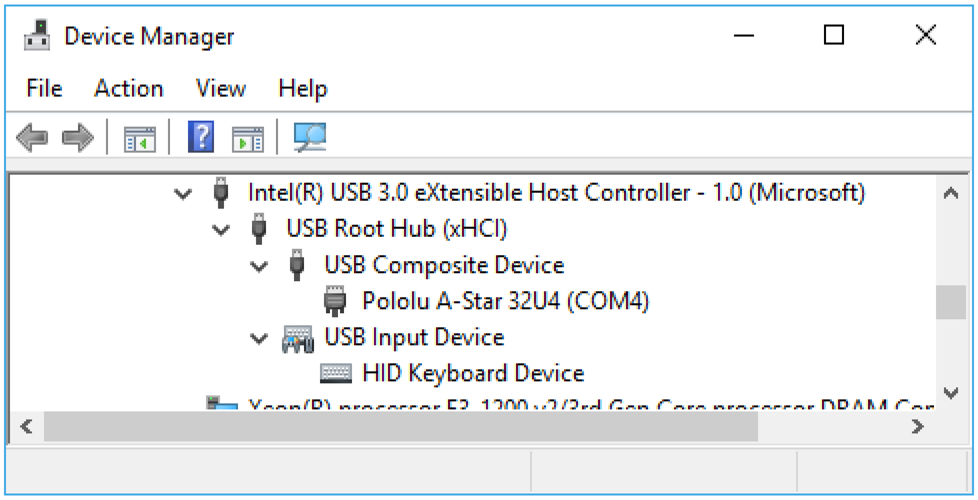
On a Windows computer, you can see the virtual serial port by going to your computer's Device Manager and expanding the "Ports (COM & LPT)" list. You should see a COM port labeled "Pololu A-Star 32U4". In parentheses after the name, you will see the name of the port (e.g. "COM3" or "COM4"). Windows will assign a different COM port number to the device depending on what USB port you plug it into and whether it is in bootloader mode or not. If you need to change the COM port number assigned to the A-Star, you can do so using the Device Manager. Double-click on the COM port to open its properties dialog, and click the "Advanced…" button in the "Port Settings" tab. From this dialog you can change the COM port assigned to the device.



Windows 10 Device Manager showing the A-Star's virtual COM port.

On a Windows computer, you can see the rest of the USB interface by going to the Device Manager, selecting **View > Devices by connection**, and then expanding entries until you find the "Pololu A-Star

32U4" COM port. Near it, you should see the parent composite device.



Intel(R) USB 3.0 eXtensible Host Controller - 1.0 (Microsoft)
USB Root Hub (xHCI)
USB Composite Device
Pololu A-Star 32U4 (COM4)
USB Input Device
HID Keyboard Device

The Windows 10 Device Manager in "Devices by connection" mode, showing that the A-Star is a composite device.

On a Linux computer, you can see details about the USB interface by running `lsusb -v -d 1ffb:` in a Terminal. The virtual serial port can be found by running `ls /dev/ttyACM*` in a Terminal.

On a Mac OS X computer, the virtual serial port can be found by running `ls /dev/tty.usbmodem*` in a Terminal.

You can send and receive bytes from the virtual serial port using any terminal program that supports serial ports. Some examples are the Serial Monitor in Arduino IDE, the **Pololu Serial Transmitter Utility** **[https://www.pololu.com/docs/0J23]**, **Br@y Terminal** **[http://sites.google.com/site/terminalbpp/]**, **PuTTY** **[http://www.chiark.greenend.org.uk/~sgtatham/putty/]**, **TeraTerm** **[http://ttssh2.sourceforge.jp/]**, **Kermit** **[http://www.columbia.edu/kermit/ck80.html]**, and **GNU Screen** **[http://www.gnu.org/software/screen/]**. Many computer programming environments also support sending and receiving bytes from a serial port.

9. The A-Star 32U4 Bootloader

Our 32U4 family of boards come with a USB bootloader that can be used in conjunction with the Arduino IDE or AVRDUDE to load new programs onto the device. This section documents some technical details of the bootloader for advanced users who want to better understand how it works. If you just want to get started using your device, it is fine to skip this section.

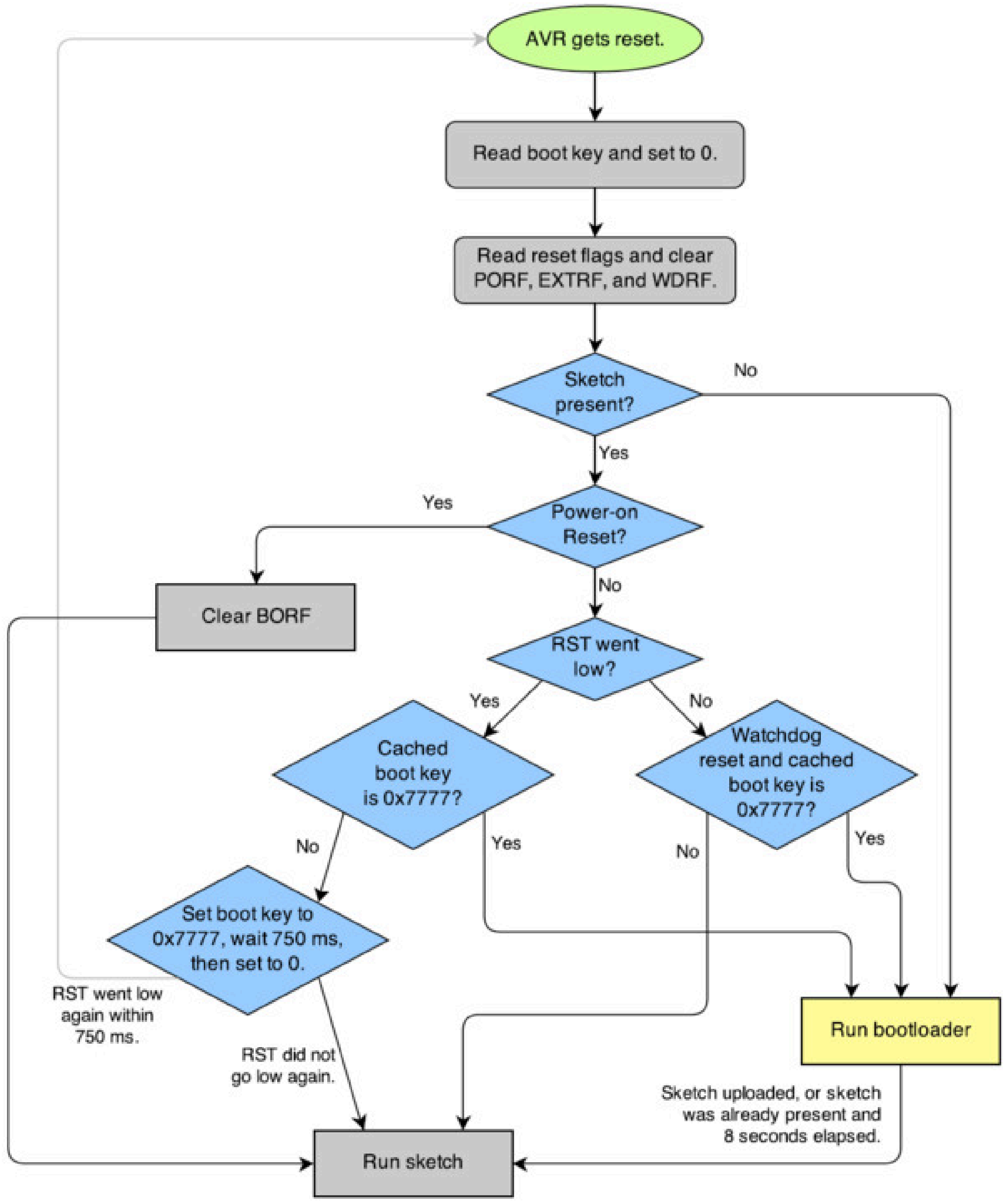
The A-Star 32U4 Bootloader is based on the **Caterina bootloader** **[https://github.com/arduino/Arduino/tree/master/hardware/arduino/avr/bootloaders/caterina]**, which is the bootloader used on the **Arduino Leonardo** **[https://www.pololu.com/product/2192]**, **Arduino Micro** **[https://www.pololu.com/product/2188]** and several other ATmega32U4 boards. The bootloader is open source and its **source code** **[https://github.com/pololu/a-star/tree/master/bootloaders/caterina]** is available on GitHub. The bootloader occupies the upper four kilobytes of the ATmega32U4’s program memory, leaving 28 KB for the user program. The bootloader’s USB interface consists of a single virtual serial port that accepts the programming commands defined in **AVR109** **[http://www.atmel.com/images/doc1644.pdf]**. The bootloader always runs first immediately after the AVR is reset.

Startup logic

The main difference between the A-Star 32U4 Bootloader and Caterina is in the startup logic. This is the part of the bootloader that runs immediately after the AVR is reset, and it decides whether to run the user program or run the rest of the bootloader. The startup logic of the Caterina bootloader is designed so that when the RST line goes low, the bootloader will run. This means that if you want to restart your program using the RST line, it will take 8 seconds before the bootloader times out waiting for an upload and the sketch starts.

The A-Star 32U4 Bootloader has different startup logic that allows you to use the RST line to reset the board with a smaller delay. If the RST line goes low once, the user program will run after a 750 ms delay. If the RST line goes low twice within 750 ms, then the bootloader will run. (This behavior is the same as on boards like SparkFun’s Pro Micro.)

The start-up logic of the A-Star 32U4 Bootloader is shown in the flowchart below:


AVR gets reset.
Read boot key and set to 0.
Read reset flags and clear PORF, EXTRF, and WDRF.
Sketch present?
No
Yes
Power-on Reset?
Yes
No
Clear BORF
RST went low?
Yes
No
Cached boot key is 0x7777?
Watchdog reset and cached boot key is 0x7777?
No
Yes
No
Yes
Set boot key to 0x7777, wait 750 ms, then set to 0.
RST went low again within 750 ms.
RST did not go low again.
Run bootloader
Sketch uploaded, or sketch was already present and 8 seconds elapsed.
Run sketch

The startup logic for the A-Star 32U4 bootloader.

Brown-out detection

Unlike many other ATmega32U4 boards, our 32U4 family of boards have brown-out detection enabled. The brown-out threshold is 4.3 V, and if the voltage on VCC goes below this then the AVR will reset.

The bootloader was designed so that the user program can detect brown-out resets. To do so, check to see if the BORF bit in the MCUSR register is set, and then clear it later. Here is some example code you could put in your `setup` function for detecting brown-out resets:

```
pinMode(13, OUTPUT);
if (MCUSR & (1 << BORF))
{
  // A brownout reset occurred.  Blink the LED
  // quickly for 2 seconds.
  for(uint8_t i = 0; i < 10; i++)
  {
    digitalWrite(13, HIGH);
    delay(100);
    digitalWrite(13, LOW);
    delay(100);
  }
}
MCUSR = 0;
```

10. Reviving an unresponsive 3pi+ 32U4

In order to load a new program onto your A-Star 32U4 device, you will need to get it into bootloader mode and send programming commands to it over its virtual serial port using appropriate software. If you are programming the device from the Arduino IDE, the sketch loaded onto the device will generally support a special USB command for putting it in bootloader mode, and the Arduino IDE sends that command automatically when you click the Upload button. However, you might find yourself in a situation where the device is unresponsive and that method will not work. This can happen for two reasons:

- You accidentally loaded a malfunctioning program onto the device that is incapable of responding to the special USB command. For example, your program might be stuck in an infinite loop with interrupts disabled.
- You loaded a program which uses a non-standard type of USB interface or no USB interface.

The following sections provide different procedures you can use to revive your device.

10.1. Reviving using the Arduino IDE

This section explains two special methods for programming an A-Star (or another of our 32U4 family of boards) using the Arduino IDE in case your usual method of programming is not working. These instructions were developed for the Arduino IDE versions 1.0.5-r2 and 1.6.0, and they might need to be modified for future versions.

Reset button

If you have an A-Star 32U4 Micro, you should connect a **momentary pushbutton [https://www.pololu.com/product/1400]** between the GND and RST pins to serve as a reset button. Other boards in our 32U4 family have a reset button you can use. Alternatively, you can use a wire to temporarily connect GND and RST together instead of using a reset button.

Resetting the board twice within 750 ms makes the board go into bootloader mode. The bootloader will exit after 8 seconds and try to run the sketch again if it has not started receiving programming commands. To revive the device, you need to make sure you start sending it programming commands before the 8-second period is over.

In bootloader mode, the yellow LED (the one labeled *LED 13*) fades in and out. It is useful to look at this LED so you can know what mode the microcontroller is in. Also, we recommend enabling verbose output during upload using the Arduino IDE's "Preferences" dialog. Looking at the LED and looking at the verbose output during the following procedures will help you understand what is going on.

The uploading-before-bootloader method

The goal of the uploading-before-bootloader method is to select a non-existent serial port in the Arduino IDE and then make sure the Arduino IDE enters the uploading phase before the microcontroller goes into bootloader mode. This method has been tested on Arduino 1.0.5-r2 and 1.6.0. This method does not work on Arduino 1.5.6-r2 because that version of the IDE gives a fatal error message if the selected serial port is not present at the beginning of the uploading phase (e.g. “Board at COM7 is not available.”).

1. Connect the device to your computer via USB.
2. In the “Tools” menu, open the “Board” sub-menu, and select “Pololu A-Star 32U4”.
3. In the “Tools” menu, open the “Port” sub-menu, and check to see if any ports are selected. If the “Port” menu is grayed out or no ports in it are selected, that is good, and you can skip to step 6.
4. Reset the board twice to get the board into bootloader mode. While the board is in bootloader mode, quickly select the new serial port that corresponds to the bootloader in the “Port” menu.
5. After 8 seconds, the bootloader will exit and attempt to run the sketch again. Wait for the bootloader to exit. Verify that either the “Port” menu is grayed out or no ports in it are selected.
6. Click the Upload button. The Arduino IDE will compile your sketch and start uploading it.
7. As soon as the large status bar near the bottom of the IDE says “Uploading…”, reset the board twice to get into bootloader mode.

The Arduino IDE will stay in the uploading phase for 10 seconds, waiting for a new serial port to appear. Once the serial port of the bootloader appears, the Arduino IDE will connect to it and send programming commands.

The bootloader-before-uploading method

The goal of the bootloader-before-uploading method is to select the bootloader’s virtual serial port in the Arduino IDE and then make sure the board is in bootloader mode at the time when the Arduino IDE enters the uploading phase.

1. Connect the device to your computer via USB.
2. In the “Tools” menu, open the “Board” sub-menu and check to see if the “Pololu A-Star 32U4 (bootloader port)” entry is visible. If this entry is visible, you can skip to step 6.
3. Using a text editor, open the file named boards.txt that provides the “Pololu A-Star” board entries. In Windows, you can typically find this file in **%LocalAppData%\Arduino15\packages\pololu-a-star\hardware\avr\**. In Linux, you can typically find this file in

~/.arduino15/packages/pololu-a-star/hardware/avr/. On macOS, you can typically find this file in **~/Library/Arduino15/packages/pololu-a-star/hardware/avr/**. If you installed the boards manually instead of using the Board Manager, you can find it in **[sketchbook location]/hardware/pololu/avr**.

4. In the boards.txt file that you opened, find the lines at the bottom of the file that start with `#a-star32U4bp` . Uncomment each of those lines by deleting the “#” character, and then save the file.
5. Close the Arduino IDE and restart it.
6. In the “Tools” menu, open the “Board” sub-menu and select “Pololu A-Star 32U4 (bootloader port)”. This entry is configured so that the Arduino IDE will send programming commands directly to selected serial port, instead of trying to send a special USB command to the port to get it into bootloader mode and then waiting for the new port to appear. By selecting this entry, the timing of the programming process below becomes easier, especially on Windows.
7. Prepare the computer to show you a list of its virtual serial ports. If you are using Windows, this means you should open the Device Manager. If you are on Linux or Mac OS X, this means you should open a Terminal and type the command `ls /dev/tty*` but do not press enter until the board is in bootloader mode in the next step.
8. Reset the board twice to get the board into bootloader mode. While it is in bootloader mode, quickly look at the list of serial ports provided by your operating system in order to determine what port the bootloader is assigned to.
9. Reset the board twice to get the board into bootloader mode again. While the board is in bootloader mode, quickly select the serial port of the bootloader in the Arduino IDE. The port can be selected in the “Port” sub-menu under “Tools”.
10. In the Arduino IDE, click the “Verify” button to compile your sketch. This could make the timing easier during the next step.
11. Press the reset button twice to get the board into bootloader mode again. As soon as you see the yellow LED fading in and out, press the Upload button.

The Arduino IDE will compile your sketch and then upload it to the selected serial port.

If the compilation of the sketch takes longer than 8 seconds, then this procedure will fail because the bootloader will time out and start trying to run the malfunctioning sketch again. If that happens, try the procedure again using a simpler sketch such as the Blink example that can be found under **File > Examples > 01.Basics > Blink**.

After reviving your device, be sure to change the Board setting back to “Pololu A-Star 32U4” and select the right Port.

10.2. Reviving using AVRDUDE

This section explains a special method for reviving an A-Star (or another of our 32U4 family of boards) using the command-line utility **AVRDUDE [http://www.nongnu.org/avrdude/]** in case your usual method of programming is not working. AVRDUDE stands for “AVR Downloader/UploaDEr”, and it is compatible with the A-Star bootloader.

If you have an A-Star 32U4 Micro, you should connect a **momentary pushbutton [https://www.pololu.com/product/1400]** between the GND and RST pins to serve as a reset button. Other boards in our 32U4 family have a reset button you can use. Alternatively, you can use a wire to temporarily connect GND and RST together instead of using a reset button.

1. Connect the device to your computer via USB.
2. Prepare the computer to show you a list of its virtual serial ports. If you are using Windows, this means you should open the Device Manager. If you are on Linux or Mac OS X, this means you should open a Terminal and type the command `ls /dev/tty*` but do not press enter until the board is in bootloader mode in the next step.
3. Press the reset button twice within 750 ms to make the AVR go into bootloader mode. You should see the yellow LED fading in and out when the AVR is in bootloader mode. While it is in bootloader mode, quickly look at the list of serial ports provided by your operating system in order to determine what port the bootloader is assigned to.
4. Type the following command in your terminal and replace COM4 with the name of the bootloader’s serial port, but do not press enter yet. This command will erase the malfunctioning program on the device but preserve the bootloader.
   ```
   avrdude -c avr109 -p atmega32U4 -P COM4 -e
   ```
5. Press the reset button twice within 750 ms to make the AVR go into bootloader mode.
6. Quickly run the command you typed previously. The command needs to be run within 8 seconds of starting the bootloader, or else the bootloader will exit and try to run the malfunctioning program again.

By following the instructions above, the malfunctioning program on the device will be erased and the device will stay in bootloader mode indefinitely. You can now load another program onto it using the Arduino IDE or AVRDUDE.

11. Related resources

To learn more about using the 3pi+ 32U4, see the following list of resources:

- The Arduino IDE has many **examples** **[http://arduino.cc/en/Tutorial/HomePage]** that can run on the 3pi+ 32U4 (although note that the control board's on-board hardware might conflict with some of these examples).
- The Arduino website has a **Language Reference** **[http://arduino.cc/en/Reference/HomePage]**, a wiki called the **The Arduino Playground** **[http://playground.arduino.cc/]**, and other resources.
- The 3pi+ 32U4 control board uses the same microcontroller as the **Arduino Leonardo** **[https://www.pololu.com/product/2192]** and **Arduino Micro** **[https://www.pololu.com/product/2188]**, so you can search the Internet for relevant projects and code examples that use one of those boards.
- **Atmel's ATmega32U4 documentation** **[https://www.microchip.com/wwwproducts/en/ATmega32u4]** has the ATmega32U4 datasheet and many related documents.
- **AVR Libc Home Page** **[http://www.nongnu.org/avr-libc/]**: this page documents the standard library of functions that you can use with GNU C and C++ compilers for the AVR.
- **Pololu3piPlus32U4 Arduino library** **[https://github.com/pololu/pololu-3pi-plus-32u4-arduino-library]**
- **Pololu3piPlus32U4 library documentation** **[https://pololu.github.io/pololu-3pi-plus-32u4-arduino-library]**
- **LUFA – the Lightweight USB Framework for AVRs** **[http://www.fourwalledcubicle.com/LUFA.php]**
- **WinAVR** **[http://winavr.sourceforge.net/]**
- **Microchip Studio for AVR and SAM Devices** **[http://microchip.com/studio]**
- **AVRDUDE** **[http://www.nongnu.org/avrdude/]**
- **AVR Freaks** **[http://www.avrfreaks.net/]**

Datasheets for some of the components found on the 3pi+ 32U4 control board are available below:

- **ATmega32U4 documentation** **[https://www.microchip.com/wwwproducts/en/ATmega32u4]**
- **Texas Instruments DRV8838 motor driver datasheet** **[https://www.pololu.com/file/0J806/drv8838.pdf.redirect]** (1k redirect)
- **ST LSM6DS33 3D accelerometer and 3D gyro module datasheet** **[https://www.pololu.com/file/0J1087/LSM6DS33.pdf]** (1MB pdf)
- **ST LIS3MDL 3-axis magnetometer datasheet** **[https://www.pololu.com/file/0J1089/LIS3MDL.pdf]**

(2MB pdf)

- **Texas Instruments TPS2113A power multiplexer datasheet [https://www.pololu.com/file/0J771/tps2113a.pdf.redirect]** (1k redirect)

Finally, we would like to hear your comments and questions on the **Pololu Robotics Forum [https://forum.pololu.com/]**!