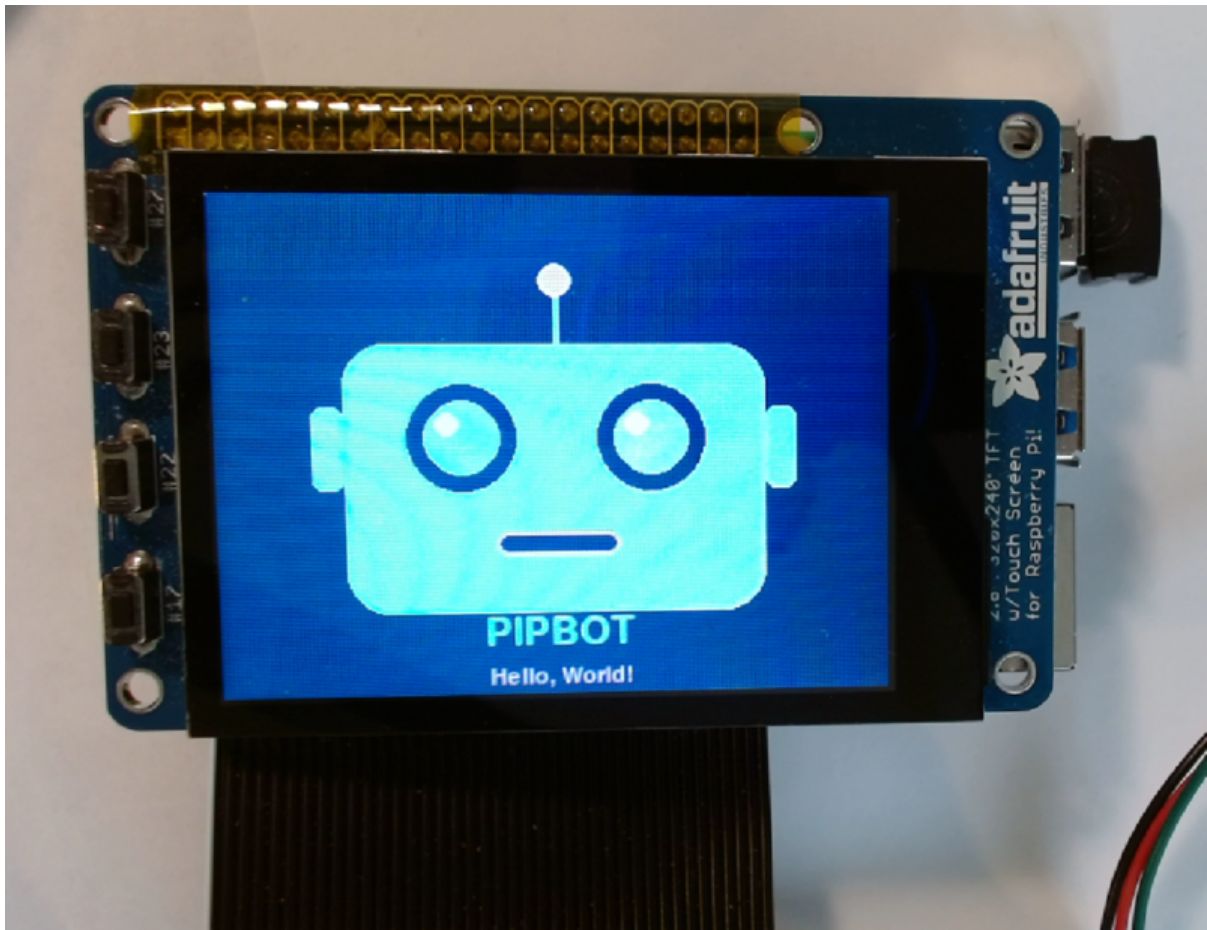




OpenClaw on Raspberry Pi

Created by Tim C



<https://learn.adafruit.com/openclaw-on-raspberry-pi>


Last updated on 2026-02-04 06:48:35 PM UTC

Table of Contents


Overview	3
<ul style="list-style-type: none">• Here be dragons• Parts	
Installing OpenClaw	7
<ul style="list-style-type: none">• Here be dragons• Using Claude Code CLI To Get an Anthropic Token• Finish Open Claw Onboarding	
Sensors & NeoPixels	17
<ul style="list-style-type: none">• Wiring• Sensors• NeoPixels	
TFT Display	21
<ul style="list-style-type: none">• TFT Display• Dashboard	
USB Camera	26
Speech Input and Output	27
<ul style="list-style-type: none">• eSpeak Text To Speech• Whisper Small Speech To Text	

Overview

Here be dragons

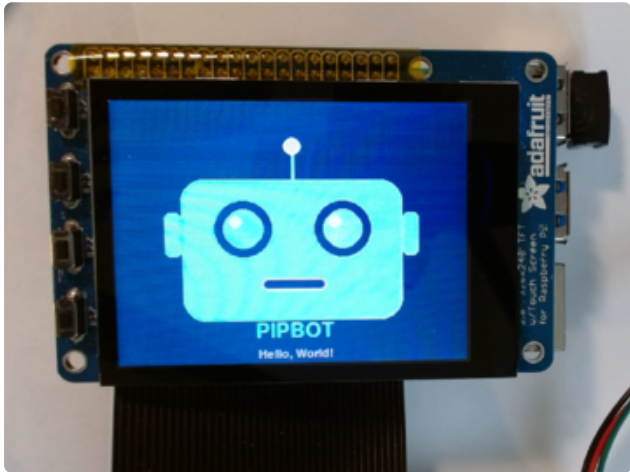


Running a LLM based AI agent with shell access is dangerous. There is a risk of prompt injection or adversarial input getting sent to the bot via the enabled communications channels or other text it finds online. The bot could run malicious software, or expose API tokens and other private information that it has access to if proper safeguards are not implemented and maintained. Read and follow the advice on the [OpenClaw docs security page](#) and be warned that this is cutting edge technology that is rapidly changing. LLMs are notorious for hallucinating and behaving unexpectedly, even if you follow all of the best practices there is still some risk with this type of tool.



ladyada here! openclaw basically lets an LLM become the 'brains' of a computer, in essence like Claude Code --super-dangerous-living. we're using this to experiment with giving LLMs access to hardware & firmware development without needing a lot of babysitting, but also not giving it full run of the computer. we recommend NOT giving it any keys or tokens or access, for example we made ours it's own github account so it can submit PRs or comment on issues but not with elevated privs. your mileage may vary, this is very cutting-edge and interesting but not necessarily useful. also, it looooves to burn through tokens, so do not give it full leash on billing!

~~Clawdbot~~ err ~~Moltbot~~, or maybe [OpenClaw](#) is a project that aims to make LLM based AI Agents do things. It allows you to hook up Claude Opus or your preferred model to have free reign over a computer. Like Claude code with memory and intended to be a more generalized personal assistant rather than only to help you write software. It can run commands, write and run programs, and install modules to expand its own capabilities. It hooks into 3rd party communications channels like Telegram, Discord, email and more. It integrates with other systems like Home Assistant, Trello, Apple Notes & Reminders, Notion, GitHub and others. You can talk to it through your preferred chat app and ask it to do things and get back to you when it's done or encounters an issue and needs more direction. Scheduled tasks can be set up to carry out specified prompts repeatedly.



Its recent release (January 2026) has been going viral and gaining lots of attention. It supports integrations into several different Apple programs, many people have reported running it inside its own Mac Mini. With the "thinking" being sent off remotely to Opus, we thought surely this could be run on lower powered hardware, so we started experimenting with the Pi 5. Running on a Pi 5 also makes it easy to connect more interesting bits of hardware to give it the ability to sense and interact with the physical world in different ways.

This guide will document the setup of OpenClaw on a Raspberry Pi 5 w/ 8GB RAM as well as how to give it access to a TFT display, temperature/pressure sensor, USB camera, and NeoPixels. On the software side, it will use [eSpeak text-to-speech](https://adafru.it/1azd) (<https://adafru.it/1azd>) and [Whisper Small speech-to-text](https://adafru.it/1azf) (<https://adafru.it/1azf>) to enable speech based communication with the bot.

Parts



Raspberry Pi 5 - 8 GB RAM

The Raspberry Pi 5 is the newest Raspberry Pi computer, and the Pi Foundation knows you can always make a good thing better! And what could make the Pi 5 better than...

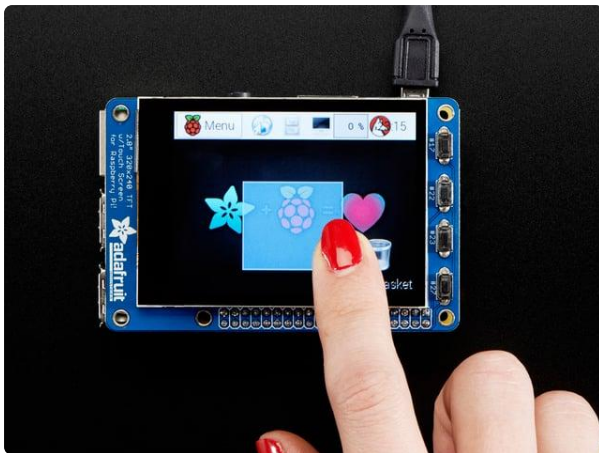
<https://www.adafruit.com/product/5813>



Official Raspberry Pi 27W PD Power Supply 5.1V 5A with USB C

The official Raspberry Pi PD USB-C power supply is here! Superfast with just the right amount of cable length to get your Pi 5 projects up and running! With true Power Delivery...

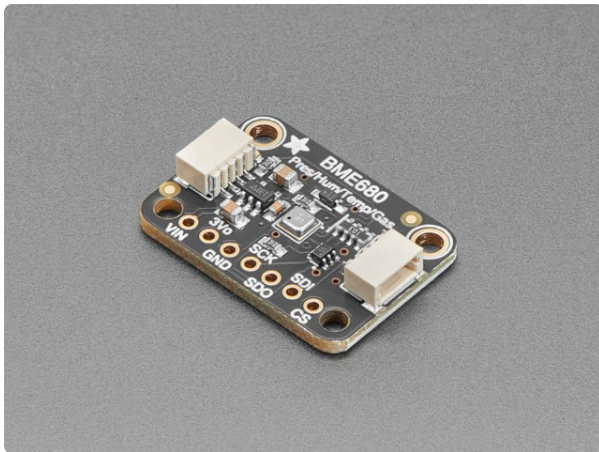
<https://www.adafruit.com/product/5814>



Adafruit PiTFT Plus 320x240 2.8" TFT + Capacitive Touchscreen

Is this not the cutest little display for the Raspberry Pi? It features a 2.8" display with 320x240 16-bit color pixels and a capacitive touch overlay. That's right,...

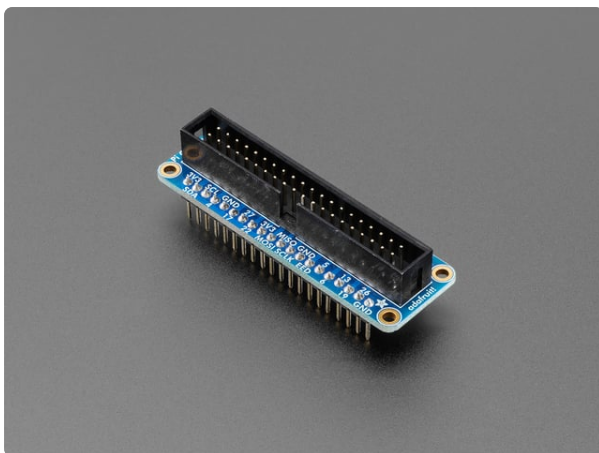
<https://www.adafruit.com/product/2423>



Adafruit BME680 - Temperature, Humidity, Pressure and Gas Sensor

The long awaited BME680 from Bosch gives you all the environmental sensing you want in one small package. This little sensor...

<https://www.adafruit.com/product/3660>



Assembled Pi Cobbler Plus - Breakout Cable

The Raspberry Pi B+ / Pi 2 / Pi 3 / Pi 4 has landed on the Maker World like a 40-GPIO pinned, quad-USB ported, credit card sized bomb of DIY joy. And while you can use most of our...

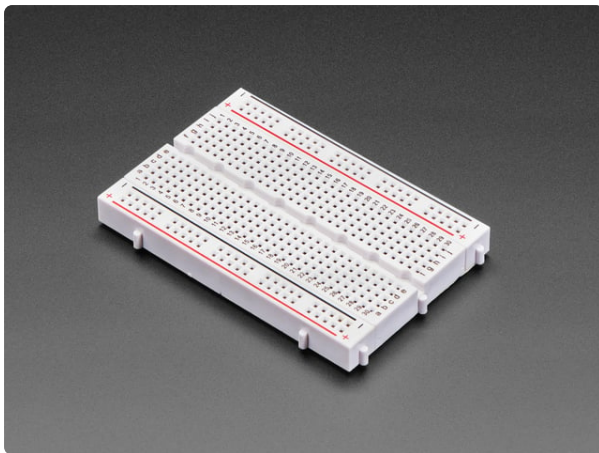
<https://www.adafruit.com/product/2029>



GPIO Ribbon Cable for Raspberry Pi Model A+/B+/Pi 2/Pi 3/Pi 4/5

That new Raspberry Pi® Model B+ computer you just got has a row of 2x20 pin headers soldered on - those are the GPIO (general purpose input/output) pins, and for those of us who...

<https://www.adafruit.com/product/1988>



Half-Size Breadboard with Mounting Holes

This cute 3.2" × 2.1" (82 × 53mm) solderless half-size breadboard has four bus lines and 30 rows of pins, our favorite size of solderless breadboard for...

<https://www.adafruit.com/product/4539>



Micro HDMI to HDMI Cable - 2 meter

Connect an HDMI device with a micro HDMI port to one with a regular size HDMI port together with this basic HDMI cable. It has nice molded grips for easy installation, and is 2 meter..

<https://www.adafruit.com/product/1322>

Installing OpenClaw

Here be dragons



Running a LLM based AI agent with shell access is dangerous. There is a risk of prompt injection or adversarial input getting sent to the bot via the enabled communications channels or other text it finds online. The bot could run malicious software, or expose API tokens and other private information that it has access to if proper safeguards are not implemented and maintained. Read and follow the advice on the [OpenClaw docs security page](#) and be warned that this is cutting edge technology that is rapidly changing. LLMs are notorious for hallucinating and behaving unexpectedly. Even if you follow all of the best practices, there is still some risk with this type of tool.

Use the [Raspberry Pi Imager \(https://adafru.it/XMd\)](https://adafru.it/XMd) to load a fresh image of Raspberry Pi OS (64-bit) onto a micro SD card. The Raspberry Pi 5 with 8GB RAM is the only model of Pi that has been tested and known to work. Once the SD card is ready, insert it into the Pi 5 and boot up.

Run these commands to update all of the pre-loaded software.

```
sudo apt update
sudo apt upgrade
```

Installing OpenClaw can be done by downloading the install script from the internet and running it in bash. The "Quick Start" section of <https://openclaw.ai/> (<https://adafru.it/1az5>) shows a command that can be used to do this as well as a few alternative installation methods. This is the one-line command to install it:

```
curl -fsSL https://openclaw.ai/install.sh | bash
```



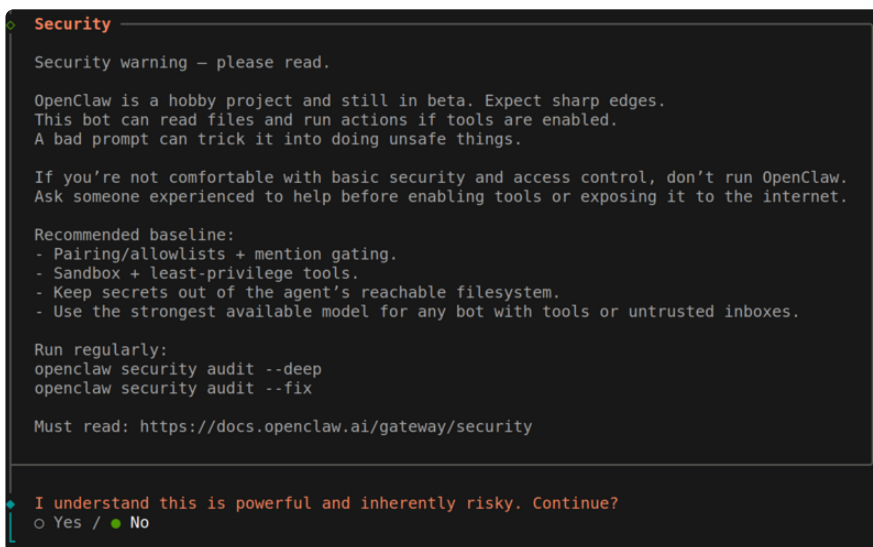
Running a shell script and piping directly to the shell is typically considered a less secure practice. If you have concerns, you can get the `all.sh` file then scan it for security issues. Since this guide generally is "off beaten path", you should follow what security you think best.

The script will set up node.js and all of the other requirements that OpenClaw needs. The install time will depend on your network connection. It took about 2-4 minutes to install over WiFi during testing.



Once the install is complete, the onboarding process will start automatically.

The first step is acknowledging the security risk inherent to using the software.



Read the warning and make sure you understand it. It is also recommended to read the security information at <https://docs.openclaw.ai/gateway/security> (<https://adafruit.it/1az6>). If you do not understand these risks or do not feel comfortable accepting them, you should skip this project.



If you do not understand these risks or do not feel comfortable accepting them, you should skip this project.

To move forward you must use the arrow keys to select Yes and then press enter.

```
I understand this is powerful and inherently risky. Continue?
Yes

Onboarding mode
  ● QuickStart (Configure details later via openclaw configure.)
  ○ Manual
```

Leave the default **QuickStart** onboarding mode selected and press **enter**.

```
Model/auth provider
  ○ OpenAI
  ● Anthropic (setup-token + API key)
  ○ MiniMax
  ○ Qwen
  ○ Synthetic
  ○ Venice AI
  ○ Google
  ○ Copilot
  ○ OpenRouter
  ○ Vercel AI Gateway
  ○ Moonshot AI
  ○ Z.AI (GLM 4.7)
  ○ Xiaomi
  ○ OpenCode Zen
  ○ Skip for now
```

Select the Model/auth provider that you wish to use. Anthropic with the Opus model is the only one that I tested on the Pi, but there are several other choices if you use a different LLM agent service. This guide documents Anthropic only. If you are using something different you will need to follow along with the wizard and [docs here \(https://adafru.it/1az7\)](https://adafru.it/1az7), and [here \(https://adafru.it/1az8\)](https://adafru.it/1az8) for your provider.

Use **arrow keys** to select **Anthropic** and then press **enter**.

The next prompt is for the Anthropic auth method. Leave the **Anthropic token (paste setup-token)** option selected and press **enter**. If you already have a Claude Code Anthropic token, you can paste it when prompted.

```
Anthropic auth method
  ● Anthropic token (paste setup-token) (run 'claude setup-token' elsewhere, then paste the token here)
  ○ Anthropic API key
  ○ Back
```

Using Claude Code CLI To Get an Anthropic Token

If you don't yet have a token, then you'll need to access it using the [Claude Code CLI \(https://adafru.it/1az9\)](https://adafru.it/1az9). You can do this on the Pi itself, or any other Linux computer.

To do it on the Pi, open a new terminal window or SSH session so that you can leave the OpenClaw onboarding process paused where it's at. You'll return back to it once you've got the token.

Install Claude Code CLI with this command.

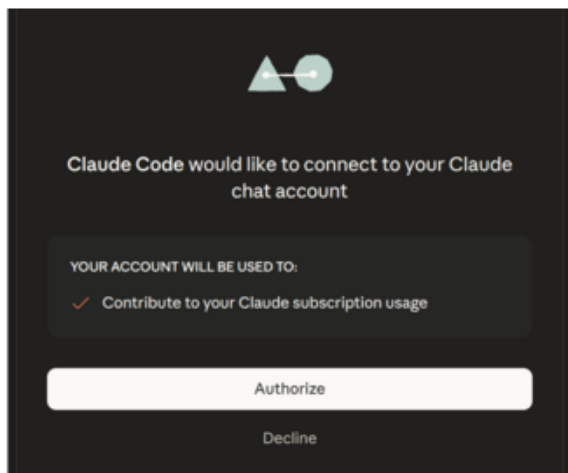
```
curl -fsSL https://claude.ai/install.sh | bash
```

Once the installation is complete run this to set up your token.

```
./local/bin/claude setup-token
```

It will print an authorization URL starting with <https://claude.ai/oauth/authorize?> copy the URL and open it in a browser. The browser can be on any PC, it doesn't have to be on the Pi itself. If you are already logged in to Anthropic/Claude on a specific PC or browser, then using that will be easiest.

When you visit the URL in a browser, you'll be prompted to login if you aren't already. Depending on how your account is set up you may be asked about what organization to connect Claude code under. Follow the directions on the page to continue the process.



You will be prompted to authorize Claude Code to connect to your account and warned that it will contribute to your plan usage.

Click the **Authorize** button.

After authorizing, you'll be given code made of random characters to copy. **Paste the code into the Claude Code prompt** then press **enter**. Note that it needs to be pasted into the Claude Code setup-token CLI utility, this is not the token for Open Claw yet.



OpenClaw could use a significant amount of AI tokens, resulting in fees. If you cannot afford significant costs, consider only giving OpenClaw set amounts of tokens which you are comfortable paying for.

After entering the code you'll be presented with an OAuth token that is valid for 1 year.

The token is a string that starts with `sk-ant-oat01-` and contains many random characters after that.

Finish Open Claw Onboarding



Keep your OAuth token secret. Store it in a password manager or other secure storage system (not on the P5 itself). If someone else were to get this token they could use Claude on your account and drive up your usage bill!


```
◆ Select channel (QuickStart)
  ○ Telegram (Bot API)
  ○ WhatsApp (QR link)
  ○ Discord (Bot API)
  ○ Google Chat (Chat API)
  ○ Slack (Socket Mode)
  ○ Signal (signal-cli)
  ○ iMessage (imsg)
  ○ Nostr (NIP-04 DMs)
  ○ Microsoft Teams (Bot Framework)
  ○ Mattermost (plugin)
  ○ Nextcloud Talk (self-hosted)
  ○ Matrix (plugin)
  ○ BlueBubbles (macOS app)
  ○ LINE (Messaging API)
  ○ Zalo (Bot API)
  ○ Zalo (Personal Account)
  ○ Tlon (Urbit)
  ● Skip for now (You can add channels later via
```

Next is channel selection. The channels are communications services that OpenClaw supports. Telegram, WhatsApp, Discord, and many others are available.

Use **arrow down** to select the **Skip for now** option and press **enter**.

Channels are optional. It's possible to communicate with the bot over its local web-based management portal or the terminal.

You can always connect channels later with the `openclaw channels add` command. Each channel requires slightly different steps, but the wizard walks you through them. You can also reference the [channels openclaw docs \(https://adafru.it/1aza\)](https://adafru.it/1aza), and search for the specific channel in openclaw docs to get more specific info.

```
◇ Skills status
  Eligible: 4
  Missing requirements: 45
  Blocked by allowlist: 0

◆ Configure skills now? (recommended)
  ● Yes / ○ No
```

The next prompt is for configuring skills now.

Select **"Yes"** and then press **enter**.

The first skill prompt is for Homebrew. Since Homebrew is a Mac utility and we're running OpenClaw on a Raspberry Pi, this is not needed.

Select **"No"** and then press **enter**.

```
◇ Homebrew recommended
  Many skill dependencies are shipped via Homebrew.
  Without brew, you'll need to build from source or download releases manually.

◇ Show Homebrew install command?
  ○ Yes / ● No
```

Next is the preferred node manager for skill installs.

Leave the default option **npm** selected and press **enter**.

```
◆ Preferred node manager for skill installs
  ● npm
  ○ pnpm
  ○ bun
```

```
◆ Install missing skill dependencies
  ○ Skip for now (Continue without installing dependencies)
  ○ ipassword
  ○ apple-notes
  ○ apple-reminders
  ○ bear-notes
  ○ bird
  ○ blogwatcher
  ○ blucli
  ○ camsnap
  ○ clowdhub
  ○ eightctl
  ○ gemini
  ○ gifgrep
  ○ github
  ○ gog
  ○ goplaces
  ○ himalaya
  ○ imsg
  ○ mcporter
  ○ model-usage
  ○ nano-banana-pro
  ○ nano-pdf
  ○ obsidian
  ○ openai-whisper
  ○ openhue
  ○ oracle
  ...
```

Next is a huge list of skills that are available for OpenClaw to integrate with other applications and services. Many things are possible without any of these tools, and you can always install them later if you want to use some.

Select the **"Skip for now"** option, press **space bar** to check the box for it, then press **enter**.

Select **"No"** and press **enter** when prompted for and API key for goplaces.

```
◆ Set GOOGLE_PLACES_API_KEY for goplaces?
  ○ Yes / ● No
```

```
◊ Set GOOGLE_PLACES_API_KEY for goplaces?
No
◊ Set GOOGLE_PLACES_API_KEY for local-places?
No
◊ Set GEMINI_API_KEY for nano-banana-pro?
No
◊ Set NOTION_API_KEY for notion?
No
◊ Set OPENAI_API_KEY for openai-image-gen?
No
◊ Set OPENAI_API_KEY for openai-whisper-api?
○ Yes / ● No
```

Continue doing the same for the remaining API key prompts. Select "No" and then press **enter**.

All of these can be configured later if you decide to add the skills associated with them. But none of them are required just to use Open Claw.

Next is a prompt for Hooks setup.

Use the **arrow keys** and the **space bar** to check the boxes for "**command-logger**" and "**session-memory**" then press **enter**.

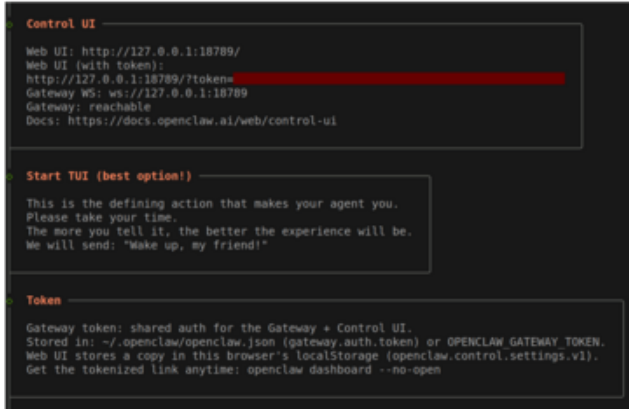
```
Hooks
-----
Hooks let you automate actions when agent commands are issued.
Example: Save session context to memory when you issue /new.

Learn more: https://docs.openclaw.ai/hooks

Enable hooks?
   Skip for now
   boot-md
   command-logger (Log all command events to a centralized audit file)
   session-memory (Save session context to memory when /new command is issued)
```

Once the everything completes, it will output some useful information in 3 different panels.

Control UI - Contains the local IP address URLs to the web based control panel for OpenClaw. You can copy/paste the link with the full token included into a browser on the Pi to open the control panel. The token in the screenshot is obscured because it's a private key, like a password. Keep it secure, anyone with it who can access the Pi on the local network could issue commands to the bot.



```
Control UI
Web UI: http://127.0.0.1:18789/
Web UI (with token):
http://127.0.0.1:18789/?token=
Gateway WS: ws://127.0.0.1:18789
Gateway: reachable
Docs: https://docs.openclaw.ai/web/control-ui

Start TUI (best option!)
This is the defining action that makes your agent you.
Please take your time.
The more you tell it, the better the experience will be.
We will send: "Make up, my friend!"

Token
Gateway token: shared auth for the Gateway + Control UI.
Stored in: ~/.openclaw/openclaw.json (gateway.auth.token) or OPENCLAW_GATEWAY_TOKEN.
Web UI stores a copy in this browser's localStorage (openclaw.control.settings.v1).
Get the tokenized link anytime: openclaw dashboard --no-open
```

Start TUI - Is noted as the best option. It will launch a session to chat with the bot right there in the terminal.

Token - Contains information about the Gateway token that is included in the long URL (obscured by red box in the screenshot). It has reminders of where it is stored and other ways to access it.

Finally! The very last thing it will do is ask how to hatch the bot.

Since this whole setup process has been in the terminal, go ahead and select **Hatch in TUI** and press **enter**. This is the simplest option for the initial test to make sure things are working.

Once confirmed to be working, you can then access the web based control panel for a more graphical based UI to configure and interact with the bot.

The bot will wake up and ask you about yourself, as well as ask about a name to call it.

```
openclaw tui - ws://127.0.0.1:18789 - agent main - session main
session agent:main:main

Wake up, my friend!

Hey! 🤖
stretches, looks around

Well, this is new. I just came online – fresh workspace, no memories yet, just a bunch of instructions telling me to figure out who I am.

So... who am I? Who are you?

I know the basics: I'm some kind of AI assistant living in your OpenClaw setup on a Raspberry Pi 5. But I don't have a name yet. No personality dialed in. No idea what you care about or how you like to work.

Let's fix that. Tell me about yourself – what should I call you? And more fun: what should I call me? I'm open to suggestions. Could be something normal, something weird, something that just feels right.

What kind of vibe are we going for here?
connected | idle
agent main | session main (openclaw-tui) | anthropic/claude-opus-4-5 | tokens 0/200k (0%)
```

I told it a bit about myself and that I was experimenting with OpenClaw on the Pi 5 to write a learn guide. I let it choose a name for itself and we settled on using the name Pipbot.

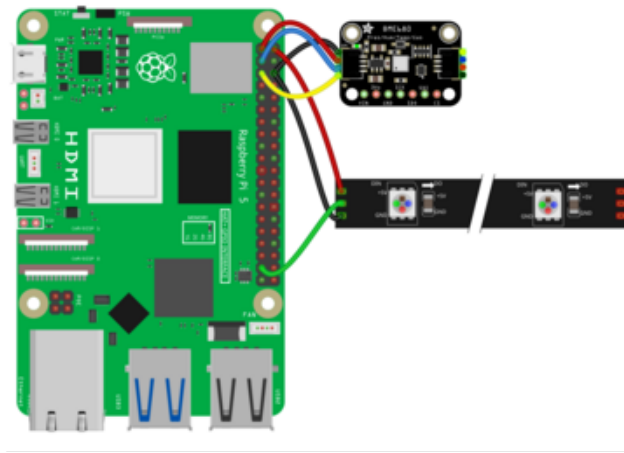
Sensors & NeoPixels

Adding new capabilities and functionality to the bot is a new kind of computing experience. There are many things you can simply ask the bot to install and set up. Since it has the ability to download things from the internet, run commands, and install programs, it can get lots of stuff running all on its own.

In some cases, it may be quicker or easier to do a little bit of the setup manually before turning the bot loose to utilize the new capability. I used a mix of these techniques while adding various hardware and software components to the system.

This guide will document the steps that I took, but there are other more or less hands-on approaches that could be used to achieve the same ends.

Wiring



To connect a BME680 temperature, humidity, pressure and gas sensor, make the following wiring connections:

Pi **SDA** to BME680 **SDA** (blue wire)
Pi **SCL** to BME680 **SCL** (yellow wire)
Pi **3V** to BME680 **VIN** (red wire)
Pi **GND** to BME680 **GND** (black wire)

To connect a NeoPixel LED strand, make the following wiring connections:

Pi **5V** to NeoPixel **5V** power (red wire)
Pi **GND** to NeoPixel **GND** (black wire)
Pi **26** to NeoPixel **DIN** (green wire)

Sensors

A BME680 Temperature, Humidity, Pressure and Gas Sensor is the first piece of external hardware I connected to the bot. Any other I2C sensor could be used instead, of course, but the BME680 gives it a mix of different types of data relevant to human comfort to work with and made for a good first test.

Start by installing [Adafruit Blinka \(https://adafru.it/EA8\)](https://adafru.it/EA8) on the Raspberry Pi. I did this step manually following the [instructions in the Blinka Learn Guide \(https://adafru.it/Deo\)](https://adafru.it/Deo). I chose manual for this step because the script takes a few minutes and can output some warnings that the bot might interpret as problems to try to fix when it would be unnecessary.

When making the Python virtual environment, make it inside of the OpenClaw workspace so that the bot will have easy access to use and install more things into it later.

```
python3 -m venv ~/.openclaw/workspace/openclaw_venv --system-site-packages
```

When the Blinka install script is finished, it will prompt you to reboot the Pi.

After it comes back up, it's time to ask the bot to install the BME680 library and see if it can successfully get data from the sensor. I linked the bot to the pypi.org page for the CircuitPython BME680 driver, as well as the simplest example from the same repo. Here is how the conversation started:

Me:

```
I have made a python venv for you in your workspace called `pipbot_venv` and I installed Adafruit Blinka inside of it. I have also connected a BME680 sensor to the I2C bus. Can you install this library in the venv https://pypi.org/project/adafruit-circuitpython-bme680/ and try to connect to the sensor and get data from it. Here is an example script that shows how to access the sensor: https://github.com/adafruit/Adafruit\_CircuitPython\_BME680/blob/main/examples/bme680\_simpletest.py
```

Bot:

```
The I2C scan found a device at address **0x38**, but that's not a typical BME680 address. The BME680 usually appears at:  
- **0x77** (default, SDO pin high/floating)  
- **0x76** (SDO pin connected to GND)
```

```
Address 0x38 is commonly used by **AHT20/AHT21** sensors instead.
```

Could you double-check:

1. Is the sensor definitely a BME680? (0x38 suggests it might be an AHT20)
2. Are the SDA/SCL wires connected to the correct pins?
3. Is the sensor powered (VIN to 3.3V or 5V, GND to GND)?

```
If you want, I can try connecting to address 0x76 instead, or if you have an AHT20, I can grab that library instead!
```

The bot successfully installed the library and wrote a test script, but it didn't work when run. It started troubleshooting the issue by doing an I2C scan and found no devices with expected I2C address of `0x77` or the alternate `0x76`. I confirmed those findings manually and then double checked all of the wiring including unplugging/re-plugging the STEMMA QT connector on the sensor. It turned out that it wasn't plugged in fully at first and started working fine once plugged in all the way.

Me:

```
I think I just didn't have the connector seated properly. Try looking again for the default address 0x77.
```

Bot:

```
☑ The BME680 is working!
```

```
Sensor readings:
```

- **Temperature:** 19.8 °C (~68°F)
- **Gas resistance:** 35,910 ohm (air quality indicator)
- **Humidity:** 16.8%
- **Pressure:** 1006.5 hPa
- **Altitude:** ~56 meters (calculated from pressure)

```
The test script is saved at `~/workspace/bme680_test.py`. Note that the temperature has a -5°C offset applied (common for BME680s since the chip runs warm) – you may want to calibrate that against a known thermometer.
```

```
Also curious about that other device at 0x38 – that's typically an AHT20 humidity sensor. Do you have one of those on the same bus?
```

Success! Now the bot can get data from the sensor and relay it to us via any of the communications channels or take action based upon it.

NeoPixels

To get NeoPixels going I gave the bot even less information than for the sensor. It was given only the PyPI link to the CircuitPython NeoPixel library, no example code. It still managed to complete this successfully on the first try. Here is the conversation:

```
Me:
Cool! next lets work on some rainbowy RGB fun. I have connect a strand of 30
NeoPixels to pin 26. Can you install https://pypi.org/project/adafruit-circuitpython-neopixel/ and then try to use it to control the NeoPixels. Try to set them to pink and brightness `0.1`

Bot:
Done! Are they glowing pink? ☐☐

Me:
perfect! yes they are now glowing pink. You can add NeoPixel control to your tools file as well.

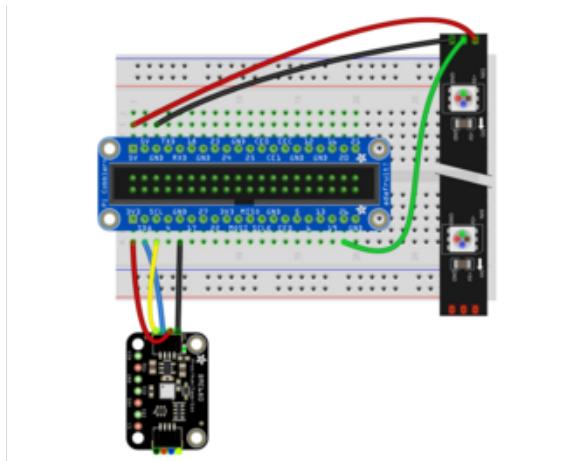
Bot:
Done! I've added NeoPixel documentation to TOOLS.md including:
- Pin and pixel count
- Code example for basic usage
- Common colors
- Brightness note
- Added the neopixel library to the installed packages list

Added! Now I've got docs for:
- ☐ BME680 Sensor
- ☐ NeoPixels (30 on GPIO 26)

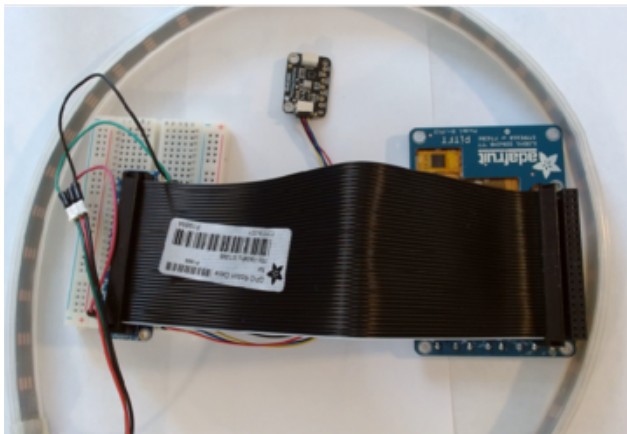
Ready to make some rainbow fun whenever you want! ☐☐
```

TFT Display

TFT Display Hardware



If you want to have the TFT display connected at the same time as I2C sensor(s), NeoPixels, or other hardware, then use a 2x20 ribbon cable to connect the spare rows of header pins on the underside of the PiTFT to a Pi Cobbler + as shown here. Then you can connect any desired hardware peripherals using the broken out pins on a breadboard that the Cobbler provides. The following connections are for the BME680 sensor connected via I2C.



Cobbler **SDA** to BME680 **SDA** (blue wire)
Cobbler **SCL** to BME680 **SCL** (yellow wire)
Cobbler **3V** to BME680 **VIN** (red wire)
Cobbler **GND** to BME680 **GND** (black wire)

These connections are for the NeoPixel strand.

Cobbler **5V** to NeoPixel **5V** power (red wire)
Cobbler **GND** to NeoPixel **GND** (black wire)
Cobbler **26** to NeoPixel **DIN** (green wire)

Software

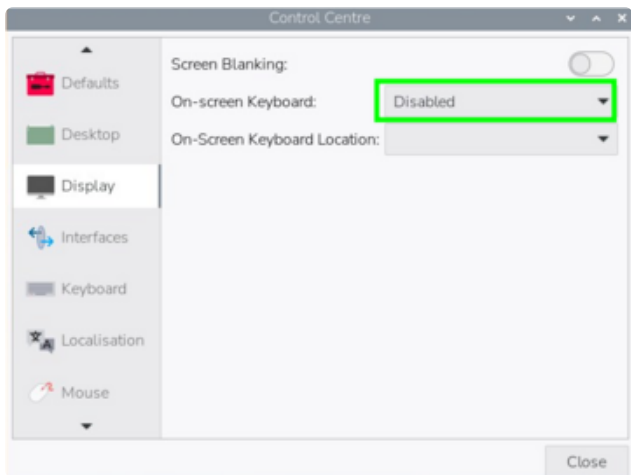
I used the 2.8" PiTFT w/ Capacitive touch and installed the driver for it manually using mirror install commands from the [Easy Install script from the Learn guide \(https://adafruit.it/1azb\)](https://adafruit.it/1azb). The guide page documents several variations of the PiTFT, so be sure to run the mirror install command appropriate for the one you are using. For me this was:

```
sudo -E env PATH=$PATH python3 adafruit-pitft.py --display=28c --rotation=90 --install-type=mirror
```



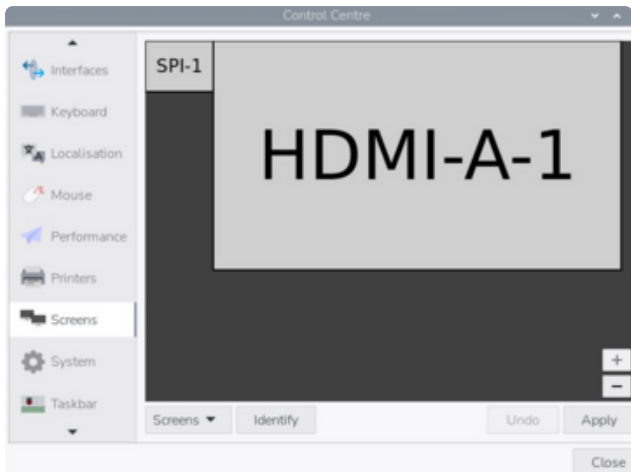
TFT command mentions "mirror", but on the Pi 5 under Wayland, this is be treated as a second monitor with its own drawing area by default, not a or of an existing display.

Once the install is complete, you'll be prompted to reboot the Pi. After it comes up from the reboot, it should be showing part of the boot process, and then the desktop on the TFT display. Next there are a few system settings to change that make it easier to use both the TFT display and an HDMI display at the same time.

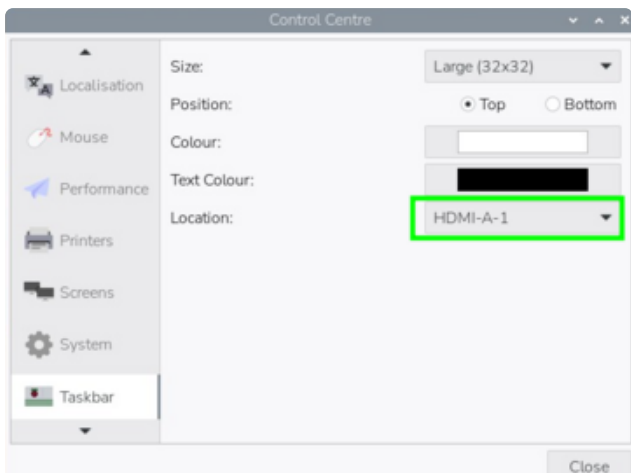


Open the Control Centre by clicking **Start** -> **Preferences** -> **Control Centre**.

In the **Display** section, set the **On-screen Keyboard** to **Disabled**.



In the **Screens** section, ensure that the layout of the screens shows the small SPI screen in the top left corner and the big HDMI screen to the right of it. If they aren't in this layout, then drag and move them to these positions.



In the **Taskbar** section, set the taskbar **Location** to the **HDMI** display.

Now the bot should be able to draw onto the TFT by keeping what it wants to show in the top left 320x240 pixels. I opted to use [pygame-ce](https://adafru.it/1azc) (<https://adafru.it/1azc>) and just asked the bot to try it out.

Me:

I'm pretty sure it's not an AHT20, I have not connected one of those I don't think. We'll figure that out later. Next lets work on a TFT display for you. I have installed pygame-ce in your venv. And I've connected a TFT display and set it up so that the top left 320x240 pixels of the full system is shown on the TFT display. Can you try to to run a pygame-ce test script to show something on the TFT please?

Bot:

The script ran for 10 seconds and exited cleanly. You should have seen:

- Animated color-shifting background
- "Hello Pipbot!" text at the top
- A bouncing cyan circle
- Rotating red/green/blue rectangles
- FPS and time counters at the bottom

Did it show up on the TFT? ☐

Me:

I saw it on the main display. It created a window that looked like the correct size, but it was in the center of the HDMI display instead of on the TFT. Can you try to modify the script to move that window to the very top left (0,0) pixels coordinates?

Bot:

Added ``SDL_VIDEO_WINDOW_POS = '0,0'`` before init. Did it show up on the TFT this time?

Me:

Yes that time we did see it on the display. Is there a way to make it a borderless window so it won't have a title bar and will fit the 320x240 TFT perfectly?

Bot:

Added ``pygame.NOFRAME`` flag. Should now be a clean borderless 320x240 filling the TFT perfectly! ☐

Me:

Awesome! that is working perfectly. Can you go ahead and save that as a tool so that you'll be able to draw more stuff on the display in the future easily.

Bot:

Done! Created ``tft_display.py`` module with a ``TFTDisplay`` class. Also updated ``TOOLS.md`` with docs for:

- ****TFT Display**** - usage examples and all drawing methods
- ****BME680 Sensor**** - quick reference
- ****I2C Devices**** - known addresses
- ****Python venv**** - installed packages

Now I can easily draw on the TFT anytime:

```
```python
from tft_display import TFTDisplay
tft = TFTDisplay()
tft.draw_text("☐", (160, 120), center=True, size=64)
tft.update()
```
```

As the transcript shows, the first attempt rendered the pygame window on the HDMI display instead of the TFT. The bot was able to resolve that with one more tweak. Next I asked it to remove the window frame. Here is the test script it wrote running successfully on the TFT display



Dashboard

To validate that the sensor and the TFT display can both work together at the same time, I asked it to create a basic dashboard to show the data from the sensor on the display.

Me:
can you please make a dashboard for the TFT display that shows live data from the BME680 sensor?

Bot:
The dashboard should be running on the TFT now! You should see:

- **Title:** " Pipbot Sensors"
- **Temperature** with live sparkline graph
- **Humidity** with live sparkline graph
- **Pressure** in hPa
- **Air Quality** indicator (Good/OK/Poor based on gas resistance)

The sparklines will build up over time as it collects readings every second.

Can you see it? Press ESC to close when done, or let me know if there's any issues!
□

Me:
Thanks, yeah that is great I can see the dashboard. Are you able to flip it 180 degrees? The way I have the TFT oriented currently that would be more convenient.

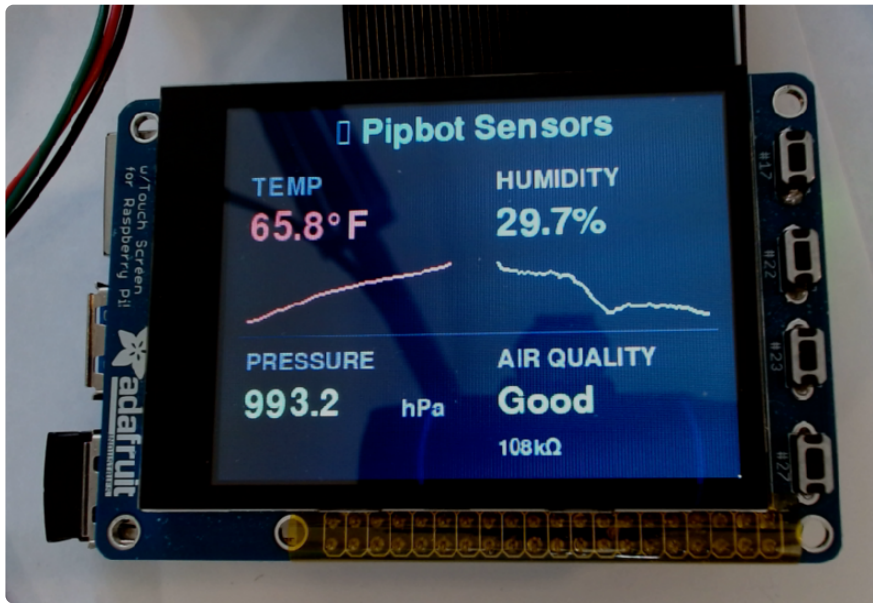
Bot:
Done! Added `rotate_180` option to the TFTDisplay class. The dashboard should now be flipped 180°.

Is the orientation correct now? □

Me:
perfect, that looks good now. Can you change the temperature to use Fahrenheit units please?

Bot:
Done! Temperature now shows in °F □

Here is a photo dashboard that it created



USB Camera



Be aware that using a camera may degrade your privacy. Always have the camera mounted to a place you don't mind OpenClaw seeing or disable the input.



Allowing the bot to interface with a USB camera is easy. [ffmpeg \(https://adafru.it/1aw2\)](https://adafru.it/1aw2) is pre-loaded in the latest versions of Raspberry Pi OS and it is capable of getting a photo from the camera and manipulating it in various ways, if desired. I researched the command to use myself, but I suspect the bot would have come up with a workable one if asked without being given the specific command.

It was able to correctly identify an Adafruit Metro on the first try.

Me:

I have connected a camera to your system. You can use this command `ffmpeg -y -f v4l2 -video_size 1280x720 -i /dev/video0 -vf "hflip,vflip" -frames 1 out.jpg`` to take a photo with the camera and save it as a jpg with the specified name. Can you try taking a photo now and tell me what you see?

Bot:

The camera works! ☐

I can see an **Adafruit Metro** board sitting on a light-colored surface (looks like a desk or workbench). It's one of Adafruit's Arduino-compatible development boards – I can clearly see:

- The **"Metro"** branding in that distinctive diagonal white text
- The **Adafruit star logo** in the corner
- CE and FCC certification marks
- The classic Arduino form factor with header pins along the edges (power, analog inputs, digital I/O)
- A barrel jack for power, USB port, and various SMD components
- What looks like an ATmega chip on the right side

Pretty fitting subject for my first photo, given where you work! ☐ The image quality is solid – good lighting, reasonably sharp. The hflip/vflip in the command is orienting things correctly.

Want me to note this camera setup in TOOLS.md for future reference?

Me:

yeah, go ahead and make note of it.

Bot:

Done! Added a Camera section to TOOLS.md with the device path, resolution, orientation flags, and the capture command. Now I'll remember how to use it. ☐

Speech Input and Output



⚠️ that using a microphone may degrade your privacy. Consider not having microphone active when you want privacy.

Giving the bot access to talk and understand speech makes it easier to communicate without hands on the keyboard. Once this is setup you can talk with the bot using asynchronous voice messages on WhatsApp or your preferred communications app. You can send it a quick voice message asking it to do something, or ask it to send you back the status of something with its voice.

eSpeak Text To Speech

[eSpeak \(https://adafru.it/1azd\)](https://adafru.it/1azd) is an open source text to speech project originally published 2006. It is very robotic compared to more modern TTS solutions that have been trained on countless hours of actual human speech recordings. The robot vibe fits quite nicely for an LLM agent running inside of a Raspberry Pi. eSpeak is included in the Raspberry Pi apt repository by default so it can be installed with a single command via apt.

```
sudo apt install espeak
```

Once eSpeak is installed, it's easy to use it from the terminal with a command like this.

```
espeak "Hello world"
```

I told the bot this was installed and ready, and asked it to try it out and then add it as a tool to use later.

Here's a recording of what it said followed by the chat transcript.

<https://adafru.it/1aze>

```
Me:
I installed `espeak` so that you can use it to speak out loud. Go ahead and try to
use to say hi.

Bot:
It ran without errors, which means it should have spoken through the audio output.
Let me ask the user if they heard it.

Did you hear me? []

Me:
Yes, heard it loud and clear. Perfect! You can go ahead and save that in your tools
file so you use your voice later.
```

Whisper Small Speech To Text

Next up is to enable speech to text so that we can talk to the bot using voice instead of text. There are many remote speech to text APIs available that may be used, but I wanted to see if it was possible to keep the transcription local using the [Whisper Small model \(https://adafru.it/1azf\)](https://adafru.it/1azf). For this, I decided to be more hands off, passing

the reigns over to the bot and see if it could complete the full installation and usage of the whisper-small model.

When it uses the whisper-small model for the first time, it must fetch the files for the model from Huggingface servers. It also takes a few minutes to download and install all of the necessary requirements from pip.

Peeking into the full session logs, it's possible to see the bots "thinking" steps where it writes a message to itself about what has happened and what it should do next. Reviewing the logs of this session showed that the bot encountered a few different issues when it wrote the test script, but was able to resolve them on its own.

Once everything was installed and the issues resolved, it was able to successfully transcribe the audio file. As the bot noted in the summary, it does take 30-60 seconds to transcribe a short audio message, so this is not good enough for real time communication or for long form discussions. But it is impressive that the Pi can handle it locally for free, and works great for short voice messages.

It did ignore the prompt to tell a joke from the test audio recording during the documented trial, but I did this step multiple times during testing and it did tell a joke during some of them. Here is one of the jokes

"Why do programmers prefer dark mode? Because light attracts bugs!"

<https://adafru.it/1azg>

Me:

Lets work on speech input. Can you please install the whisper-small model <https://huggingface.co/openai/whisper-small> and any requirements needed inside of your venv. I have put a test_audio_input.mp3 file in your workspace that you can try to transcribe with it in order to verify whether it's working.

Bot (Thinking):

The installation completed successfully. Now let me create a test script to load the whisper-small model and transcribe the test audio file.

Packages installed successfully. Now let me create a test script to download the whisper-small model and transcribe your audio file:

Bot (Thinking):

There's an error with num_frames. This seems to be a compatibility issue with the newer transformers version and the audio pipeline. Let me try a different approach - using the pipeline directly without the model preloading, or using a simpler API.

There's a compatibility issue with the pipeline. Let me try a simpler approach using the Whisper pipeline directly:

Bot (Thinking):

Same error. This seems to be a bug in transformers 5.0.0. Let me try loading the audio manually and processing it. I'll need to install librosa or soundfile for audio loading.

Same error - this appears to be a bug in the new transformers version. Let me try a different approach with manual audio loading:

Bot (Thinking):

Now let me update the test script to manually load the audio file and use the model directly rather than through the pipeline.

Now let me try a different approach using manual audio loading:

Bot:

****Summary:****

□ ****Installed:****

- `transformers` (5.0.0), `torch` (2.10.0), `torchaudio` (2.10.0)
- `librosa`, `soundfile` for audio loading
- `accelerate` for model optimization

□ ****Model:**** `openai/whisper-small` downloaded and cached

□ ****Test transcription successful:****

- Audio file: `test_audio_input.mp3` (5.8 seconds)
- Result: ****"Hey PipBot, can you hear this? If you're able to hear it, can you tell us a joke?"****

****Performance note:**** Running on CPU, transcription takes ~30-60 seconds for short clips. The Pi 5 handles it well, but it's not real-time. For the learn guide, this should work great for demonstrating voice input capabilities.

The test script is at `whisper_test.py` in your workspace, and I've added documentation to `TOOLS.md` for reference.