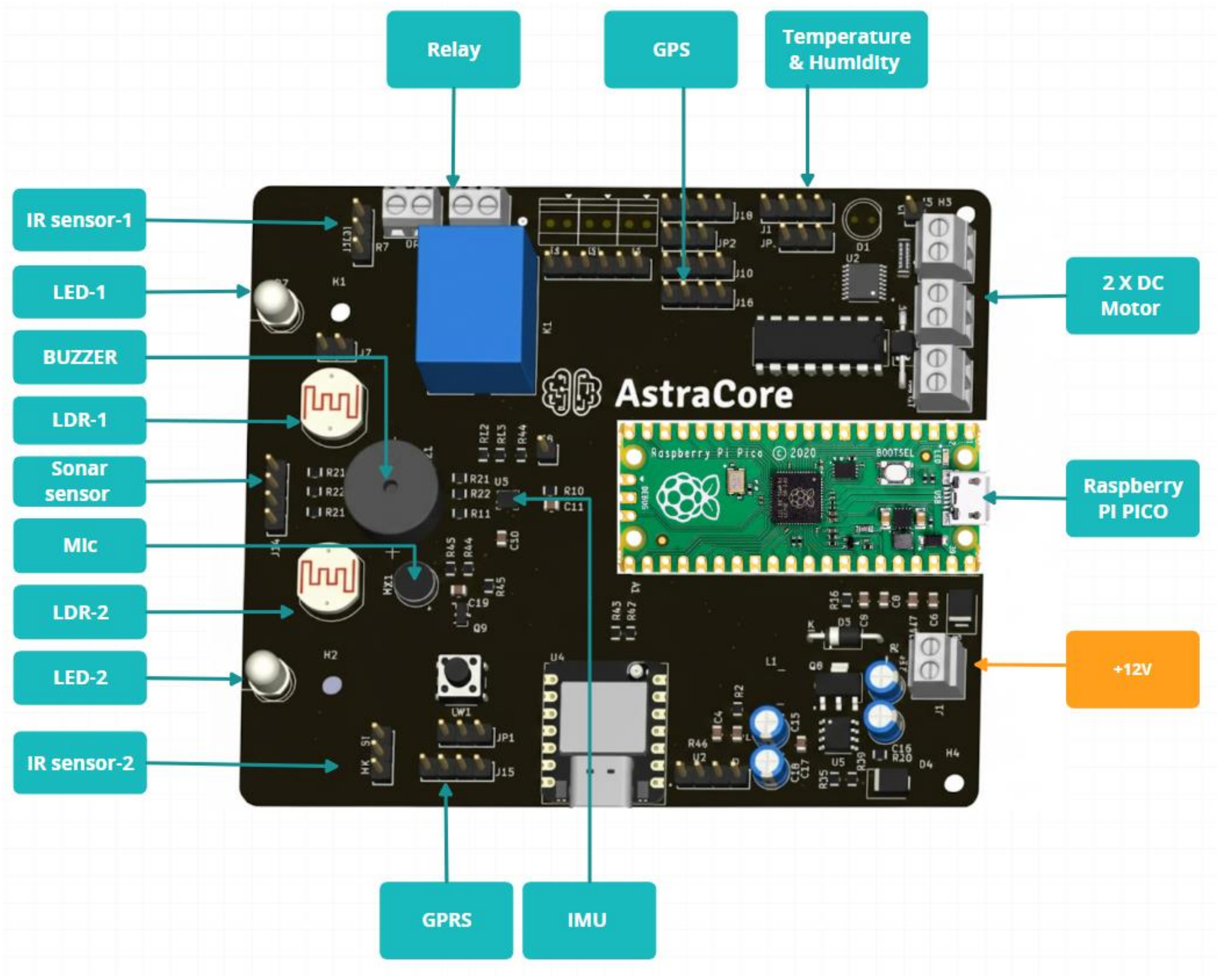
Astracore

Table of Content:

1. Introduction

Astracore is an educational platform designed to inspire and engage young learners. Equipped with a variety of built-in hardware, it provides hands-on learning experiences in electronics and programming. This platform helps young minds grasp fundamental concepts while preparing them for the future of AI and digital transformation

The Raspberry Pi Pico supports block-based and MicroPython programming IDEs like **MicroBlocks or Thonny**. This allows beginners to start programming with no prior knowledge. Block programming is a drag-and-drop method where users can simply drag and drop pre-programmed blocks into the IDE's project area, making it an easy and intuitive way to learn Coding.



Key Features:

1. Processor board compatibility
   a. Raspberry PI PICO
   b. Raspberry PI PICO W
   c. Seed studio XIAO boards
2. Motor driver output
3. 2 x LEDs
4. BUZZER
5. 1 x SPST Relays
6. 1 x Tact switch
7. Microphone
8. Inertial Measurement Unit (IMU)
9. 2 x LDR
10. Power supply: USB powered or DC source powered

External Interfaces

1. IR sensor interface
2. Sonar sensor interface
3. 3 x I2C interface
4. 2 x UART interface

2. Ordering information

Variant name	Description	Raspberry PI controller
Astracore Basic	Astracore controller Board	Raspberry PI PICO
Astracore Advance	Astracore controller Board with Sensors	Raspberry PI PICO
Astracore Advance +	Astracore controller Board with Advance Sensors	Raspberry PI PICO 2W

Features Matrix

Features	Astracore Basic	Astracore Advance	Astracore Advance +
Processor	✓	✓	✓
Buzzer	✓	✓	✓
Mic	✓	✓	✓
LDR	✓	✓	✓
2x LED	✓	✓	✓
1x Relay	✓	✓	✓
IMU	✓	✓	✓
Tact switch	✓	✓	✓
2xMotor and wheel		✓	✓
2 x IR sensor		✓	✓
Sonar sensor		✓	✓
Humidity sensor			✓
Wifi			✓
BLE			✓
GPS			✓
GPRS			✓

3. Bot- Dos & Don'ts

✅ DOs – Safe & Recommended Usage

- Handle Carefully
  - Hold the board only by the edges.
  - Use anti-static precautions when possible.
- Use Proper Power Supply
  - Supply 12V only at the power terminal.
  - Use USB only for programming or low-power operation.
- Check Connections Before Powering
  - Ensure daughter boards, sensors, and cables are properly aligned.
  - Double-check polarity on power cables and motors.
- Mount on a Chassis
  - Place the board on the provided acrylic chassis before operation.
  - Prevents accidental shorts.
- Use Correct Ports
  - Connect sensors to their labeled headers (IR, LDR, Sonar, GPS, IMU, Motor Driver, etc.)
- Test Modules One at a Time
  - When learning or debugging, test each peripheral (IR, LDR, relay, motor, GPS) individually.
- Update Code Safely
  - Save programs as **main.py in raspberry pi pico** only after testing to run independently from battery
  - Use Thonny IDE or recommended programming tools.
- Keep the Board Clean
  - Dust-free, dry environment improves longevity.
- Store Properly
  - Keep unused daughter boards in pouches.

❌ DON'Ts – Avoid Damage or Malfunction

- Do Not Touch Electronic Components frequently to avoid static discharges.
  - Avoid touching pins, chips, or solder pads with bare hands.
- Do Not Power with Incorrect Voltage
  - Supplying more than 12V can permanently damage the board.
  - Never connect multiple power sources simultaneously.
  - Never connect to AC (230V available at home)voltage directly
- Do Not Force Connectors
  - Never push headers in the wrong orientation.
  - Do not bend or twist the pins.
- Do Not Place on Metal Surfaces
  - Conductive surfaces may short the board.
- Do Not Run Motors Directly From the CPU

   - Always use the onboard motor driver.
- Do Not Disconnect Modules While Powered
   - Hot-swapping can damage sensors or the PCBA.
- Do Not Overload Output Pins
   - Avoid connecting high-current devices directly to GPIO without drivers.
- Do Not Expose to Water or Heat
   - Keep away from liquids, direct sunlight, or high temperatures.
- Do Not Modify the Board
   - Avoid soldering extra components unless specified.
   - Unauthorized modifications void protection and may damage circuitry.

4. Unpacking instructions

1. Safety & handling

• Do not connect the board to AC supply(230V available at home) directly

• Do not power the board before completing the basic assembly.

2. Assembly Manual & Product Information

• The first item you should locate in the package is the Assembly Manual(inside kit) and Product Information booklet(available online).

• Keep this document aside for immediate reference during unpacking and setup.

3. Items included in the package

Verify that your kit contains the following items:

1. AstraCore PCBA (Main Development Board)
2. CPU Board (Pre-installed on the AstraCore PCBA)
3. Acrylic Chassis
4. Daughter Boards (Sensor/Peripheral Modules)
5. Cable Assemblies (Sensor Cables, Power Cables, Jumpers, Connectors)
6. Assembly Manual

4. Unpacking steps

AstraCore (Acore) PCBA

• Remove the anti-static bag and hold the board by its edges.

CPU Board (Pre-installed)

• Verify that the CPU module is securely seated on the Acore board’s header pins.

• In case of CPU board removal, ensure the pin-1 mating while inserting back.

• Avoid touching pins or contacts directly.

Acrylic Chassis
• Remove any protective films on both sides of the acrylic plates.

Daughter Boards
• Remove each board carefully from its protective pouch.

Cable Assemblies
• Take out all cables and connectors.

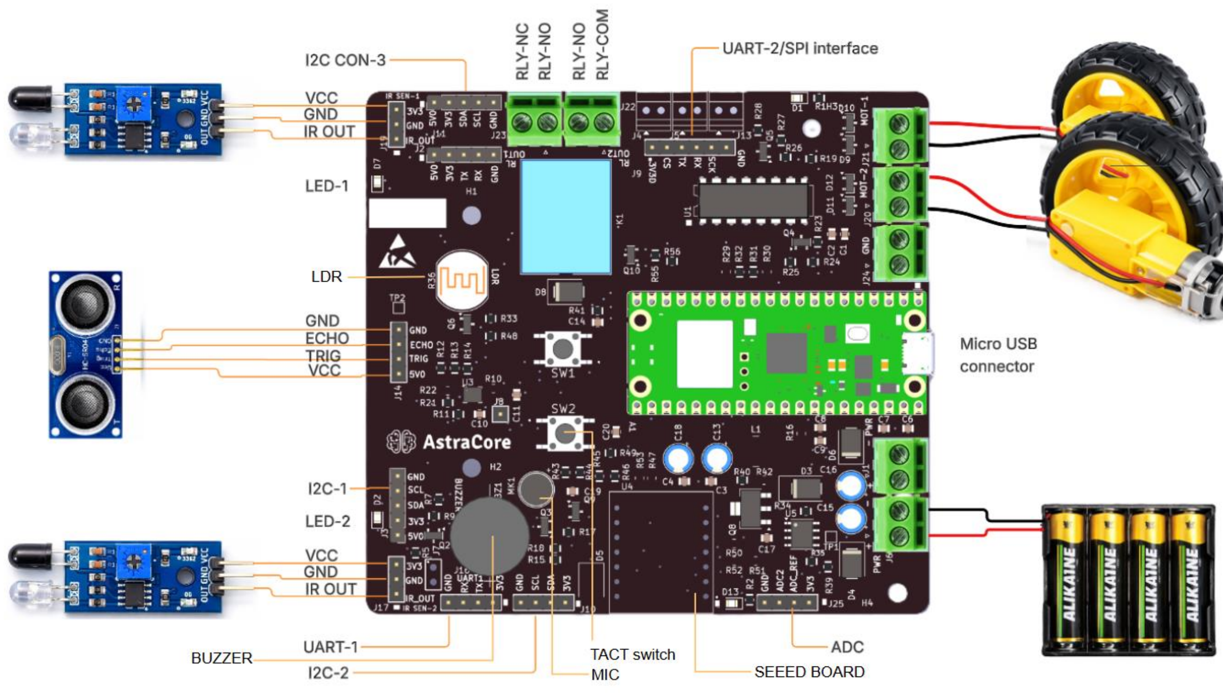
Final Verification
• Cross-check all items with the "Items Included" list and the Assembly Manual.
• In case of missing or damaged items, notify customer support.

Storage & Handling
• Keep unused electronic modules inside their anti-static pouches.
• Retain the packaging box for future storage or transportation.

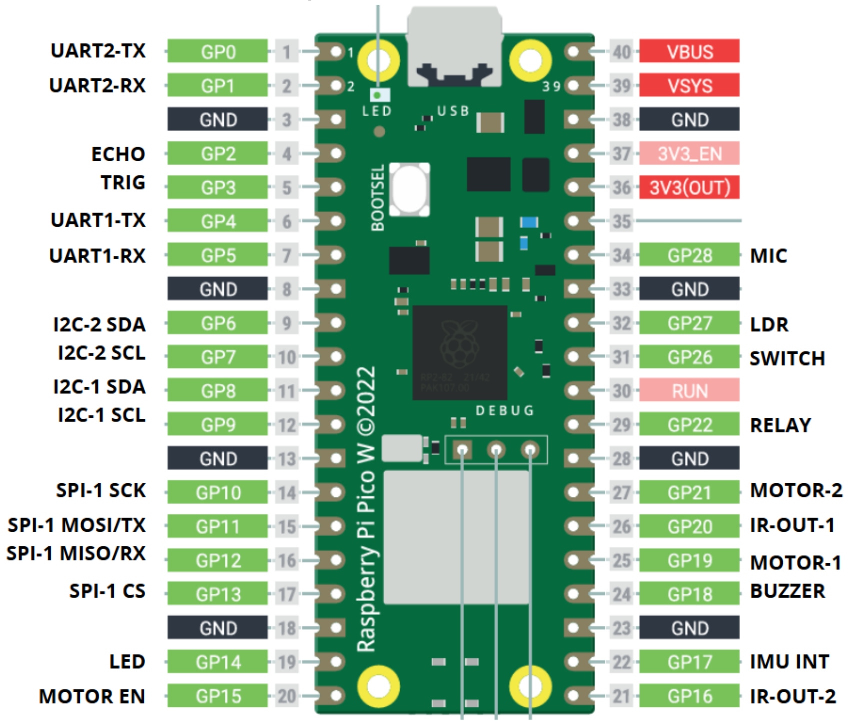
Your AstraCore kit is now ready for assembly and project development.

Peripheral Interface information

Connectors	Features
On board	Buzzer
On board	Mic
On board	LDR
On board	2x LED
On board	1x Relay
On board	IMU
J20,J21,J24	2xMotor and wheel
J17,J19	2 x IR sensor
J14	Sonar sensor
J12	Humidity sensor
Pico Board	Wifi
Pico Board	BLE
J10	GPS
J15	GPRS



INTERFACE TO PICO(GPIO PIN NUMBER FOR PROGRAMMING)



5. Toddling with Scratch

Fresh minds are always eager to build things on their own and see quick, visible results. That's why Scratch is a crucial part of this curriculum.

Scratch is the world's largest coding community for children and a beginner-friendly programming language with a simple visual interface. It enables young learners to grasp programming concepts through block-based coding. Scratch is Developed, designed, and moderated by the nonprofit Scratch Foundation, Scratch fosters creativity and computational thinking in an engaging way.

5.1. Where is the Scratch

Before we start, where can we find the Scratch app..?

The Scratch application is available in web application also desktop application. For Web applications it is not necessary to install any application on the laptop, just type the web page address and play init. Those want the offline application reach to the download page and install on the laptop

1. Online Version (Web Application) The web-based version offers immediate access without any installation:

- Simply open your web browser
- Navigate to scratch.mit.edu
- Begin programming instantly
- Requires only an internet connection
- Perfect for quick access and cloud storage of projects

2. Offline Version (Desktop Application) For users preferring local access, Scratch offers a downloadable desktop version:

- Compatible with multiple operating systems:
- Microsoft Windows
- Apple macOS
- Linux distributions
- Installation process:
- Visit scratch.mit.edu/download
- Select your operating system
- Download the appropriate installer
- Follow the installation wizard

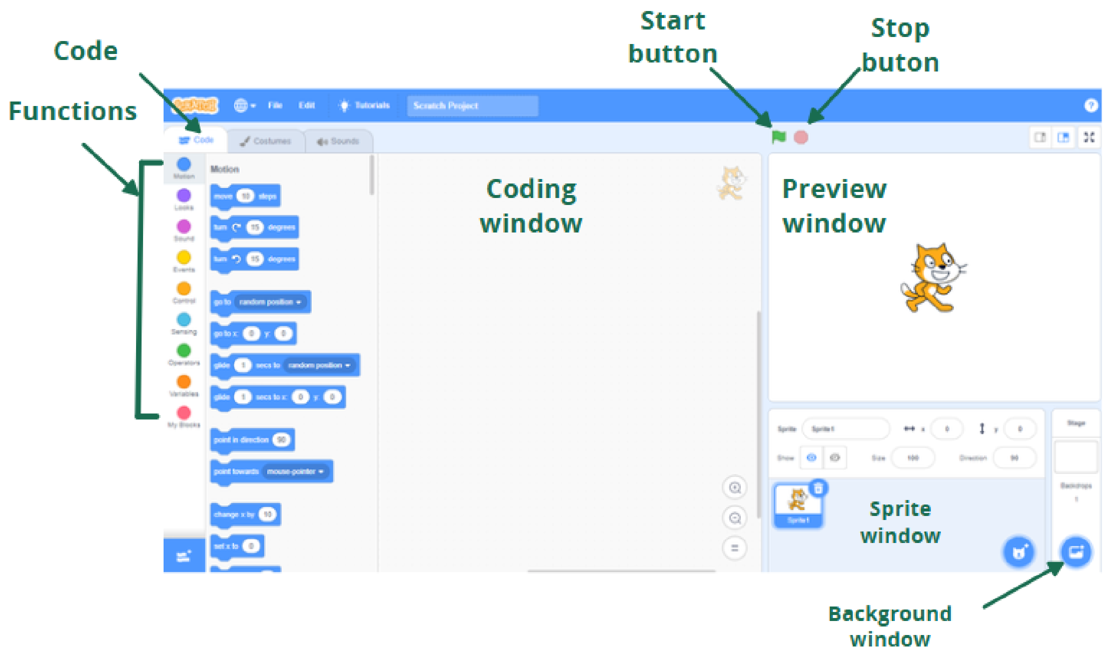
The web version is ideal for beginners and those with stable internet connections, while the desktop version suits environments with limited connectivity or those requiring offline access. Both versions offer identical features and programming capabilities, ensuring a consistent learning experience regardless of your chosen platform.

5.2. Getting started:

Imagine playing a car racing video game where upgrading a car is as simple as selecting an item from the upgrade gallery, and then dragging and dropping it onto the upgrade screen. The user can instantly see and visualize the changes being made.

Similarly, in block programming, the user can pick functions from the code window and drag them to the console. A play and stop button allows them to run and halt the function as needed.

To enhance understanding, it is essential to explain the Scratch program console



Code: This is where various functions are available. From the code window, you can select and use the required functions.

Functions: In Scratch, different actions such as motion, control, and sensing are available. These actions can be selected from the function window and dragged into the coding window.

Sprites: Sprites are 2D images used to experiment with different functions. By default, a cat sprite appears, but you can assign functions to any sprite to animate it.

Background: Choose a suitable background that complements the selected sprite.

Start & Stop Button: Once the required functions are arranged in the desired sequence, click the Start button to run the code or the Stop button to halt execution.

Preview Window: As the name suggests, the Preview window displays the output when the Start button is pressed, executing the selected functions accordingly.

Detailed User information can be found in [scratch wiki]

Get Started with Scratch: https://scratch.mit.edu/projects/editor/?tutorial=getStarted

Experiments:

1. Sprite & background
2. Sprite Move from right to left
3. Talking Sprite
4. Make the Sprite to move 10 steps when click on the flag/start button

5.3. **Basics for embedded programmers:**

- motion

The Scratch programming platform can be utilized by different groups with various interests, such as gaming, classroom learning, and as a tool for young programmers to easily understand programming logic.

The essential code sections that a beginner should learn from Scratch are:

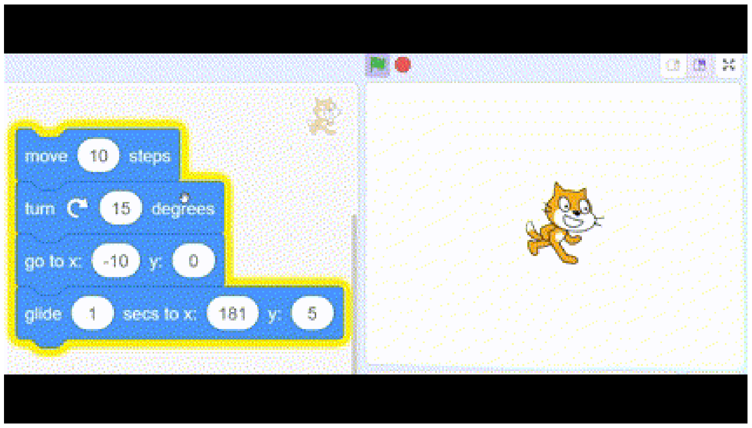
1. Motion
2. Events
3. Control
4. Operators
5. Variables

- **Motion Blocks**

Motion blocks help control the movement of objects or sprites. You can make them move, turn, glide, or go to a specific position. In Scratch, **sprites** are the characters or objects that you want to move.

Example 1 Walking Cat: Exercise to Learn Move, Turn, Glide, and Go to Position

1. Drag the required blocks in the order given below.
2. Align the blocks properly.
3. Click on the blocks and observe the movement of the sprite.



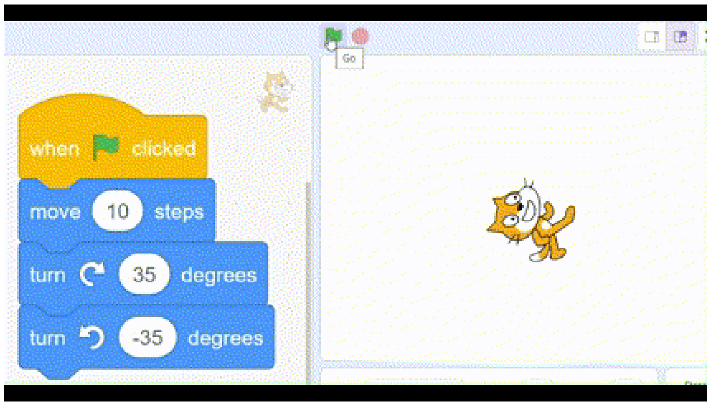
```
move (10) steps
turn ↻ (15) degrees
go to x: (-10) y: (0)
glide (1) secs to x: (181) y: (5)
```

- **Events Blocks - Start the Action!**

As the name suggests, Event blocks act like the 'go' button! They trigger your program to start when something happens — for example, when you click the green flag, the program begins running

Example: Spinning Cat: Exercise to Learn Events

1. Drag the Event & other required blocks for the dancing cat as given below
2. Click on the green flag to start the sprite to dance

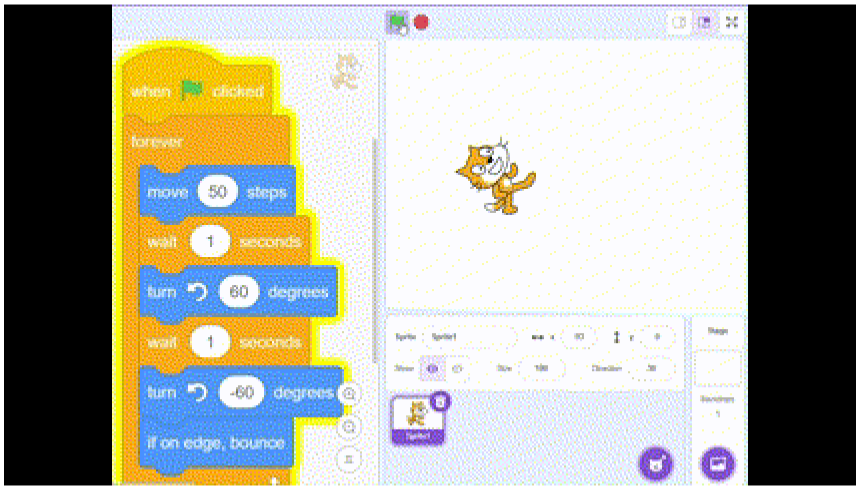


- **Control Blocks**

Control blocks help the user define action boundaries with specific actions or conditions. In these actions, there could be a count or a number that determines how many times an action is performed. For example, the "wait" block is used with a time value, such as "wait 10 seconds," meaning the sprite will wait for 10 seconds before performing the next action. Similarly, loops repeat the action specified inside them until a condition is satisfied, at which point the loop exits.

Example: IN LOOP loop Cat:

1. Drag the Control functions like the "Forever" loop and "Wait" block in the order given below.
2. Add an event block "When Green Flag Clicked" to run the sprite.



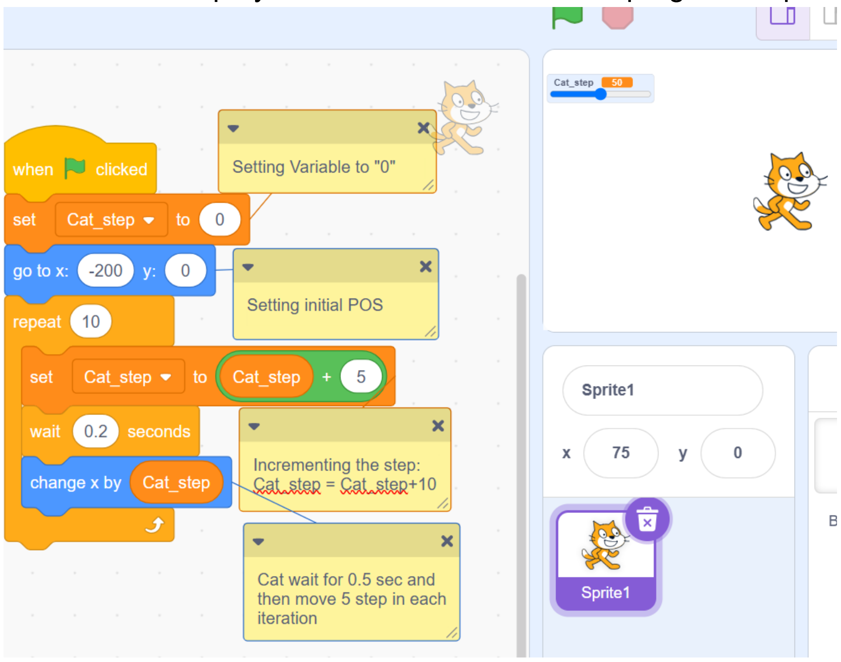
```
when [flag] clicked
forever
    move (50) steps
    wait (1) seconds
    turn ⟲ (60) degrees
    wait (1) seconds
    turn ⟲ (-60) degrees
    if on edge, bounce
```

- **Operators and Variables**

  Operators are blocks used to perform mathematical calculations, compare values, and work with logic (true/false decisions).Operators are used to **control the behaviour** of Sprites by program to make in side conditions along with integers, strings or variable to make decisions, do calculations, and create more intelligent responses on the sprite actions.

Variables , The word itself says the nature of it. Variable are used to store a specific value for that moment and can be changed during the course of the program using operator .

For example , if we want to move the cat 5 steps in every loop, we can create a variable and increment the step by a count of 10. Refer below program steps



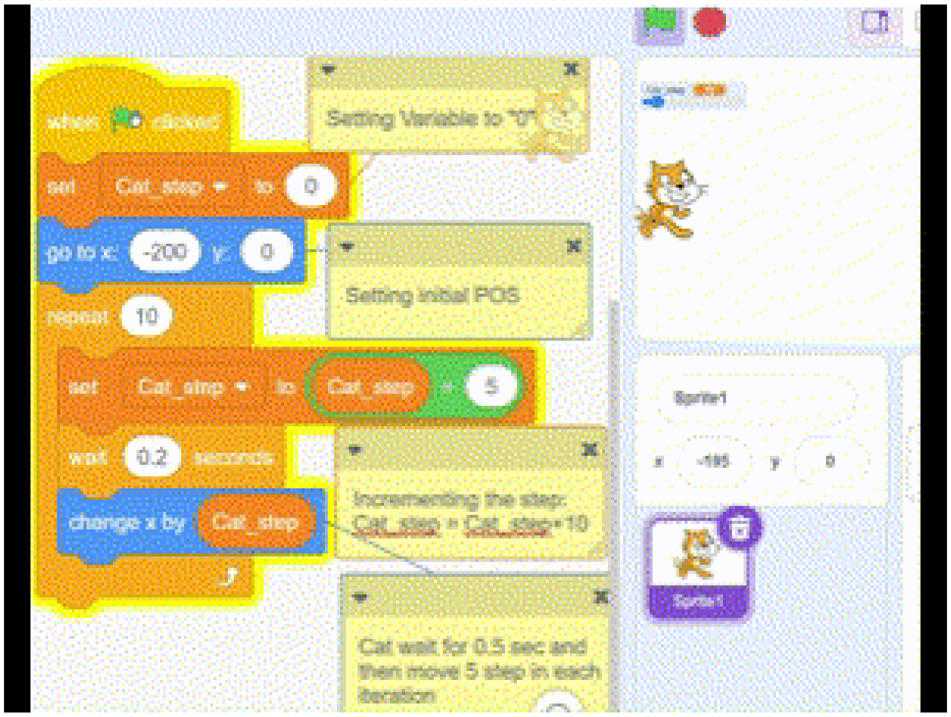
```
when [flag] clicked
set [Cat_step] to (0)
go to x: (-200) y: (0)
repeat (10)
    set [Cat_step] to ((Cat_step) + (5))
    wait (0.2) seconds
    change x by (Cat_step)
```

Setting Variable to "0"

Setting initial POS

Incrementing the step: Cat_step = Cat_step+10

Cat wait for 0.5 sec and then move 5 step in each iteration



```
when 🏴 clicked
set Cat_step ▾ to 0
go to x: -200 y: 0
repeat 10
    set Cat_step ▾ to (Cat_step + 5)
    wait 0.2 seconds
    change x by Cat_step
```

Setting Variable to "0"

Setting initial POS

Incrementing the step:
Cat_step = Cat_step+10

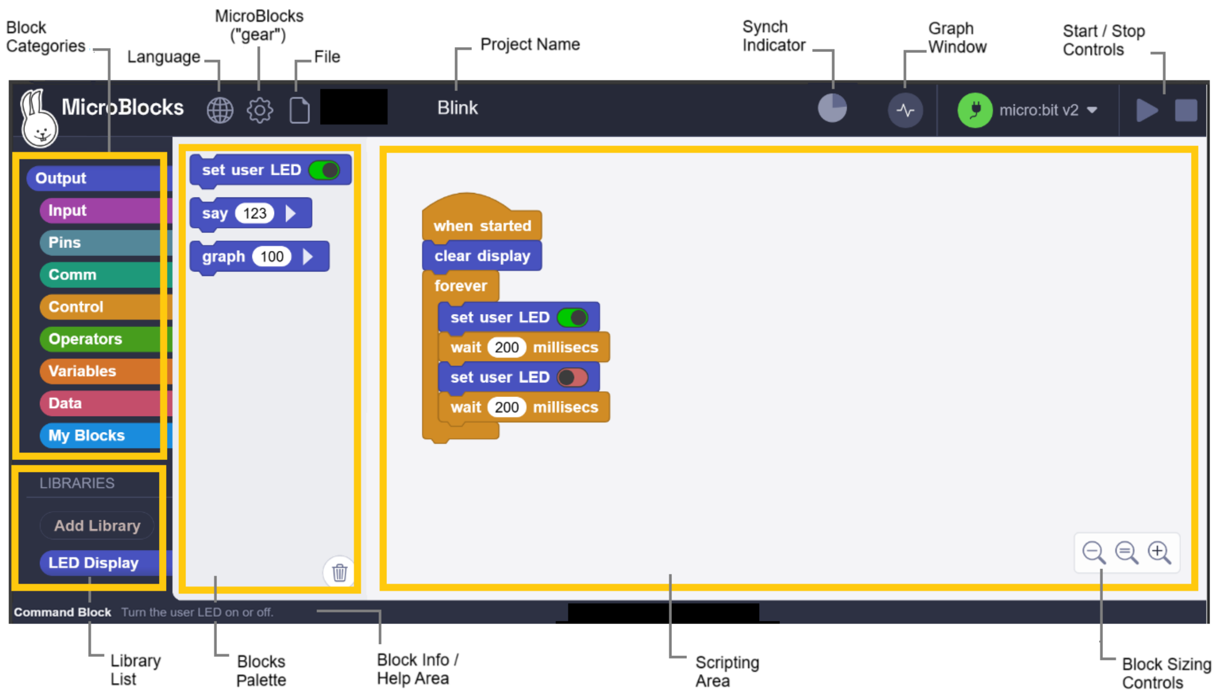
Cat wait for 0.5 sec and then move 5 step in each iteration

6. Getting Started with with Microblocks

6.1. Microblocks and Scratch-3

The Microblock (free, block-based programming software) is similar to Scratch-3 platform and both are block based programming . Scratch is focussed on computer based programming education, which helps kids to learn the basics of programming easily. Where as the Microblock is meant for embedded programming and program which can real time execution on raspberry pi pico board. Following section will brief how to connect, use and build application using microblock along with Astarcore platform. A more detailed link on how to use Microblock in general is available on below wiki page from microblock https://wiki.microblocks.fun/en/ide

The picture below shows the principal areas of the MicroBlocks editor



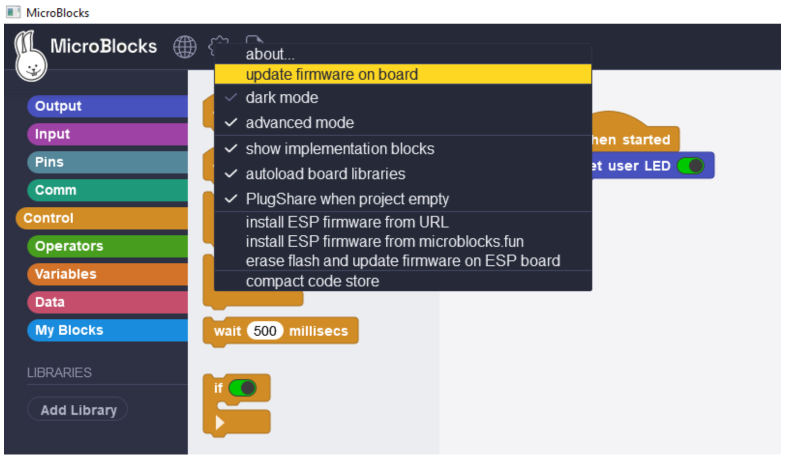
6.2. Connect to microblock

Step 1: Download and Install MicroBlocks

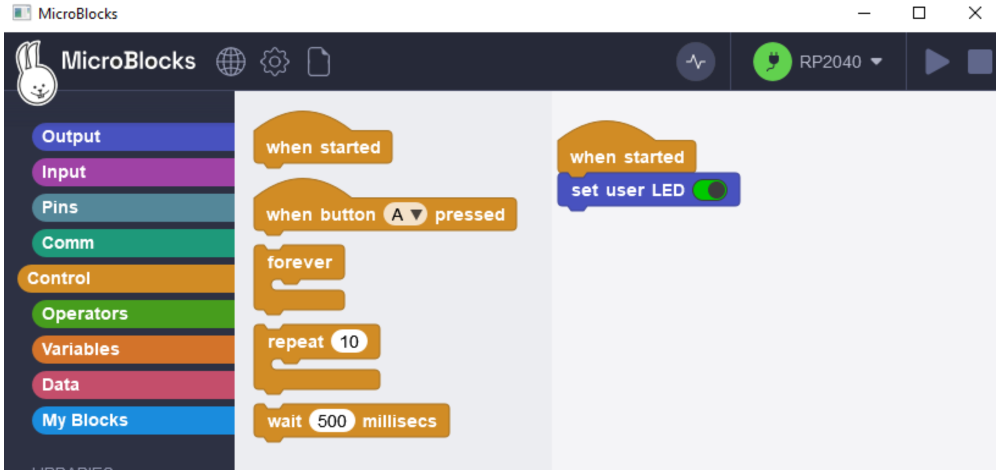
- Go to https://microblocks.fun
- Click Download
- Choose the version for your operating system
- Install and open the MicroBlocks application

Step 2: Connect Raspberry Pi Pico to Your Computer

- Plug one end of the USB cable into the Raspberry Pi Pico
- Plug the other end into your computer
- Make sure the cable supports data transfer (not charging-only)
- Put Raspberry Pi Pico into Boot Mode (First Time Only) by Press and hold the BOOTSEL button on the Pico while connecting to USB port
- Release the button after connecting
- A new drive named RPI-RP2 will appear on your computer
- Go to **Gear**→ **Update firmware on board**



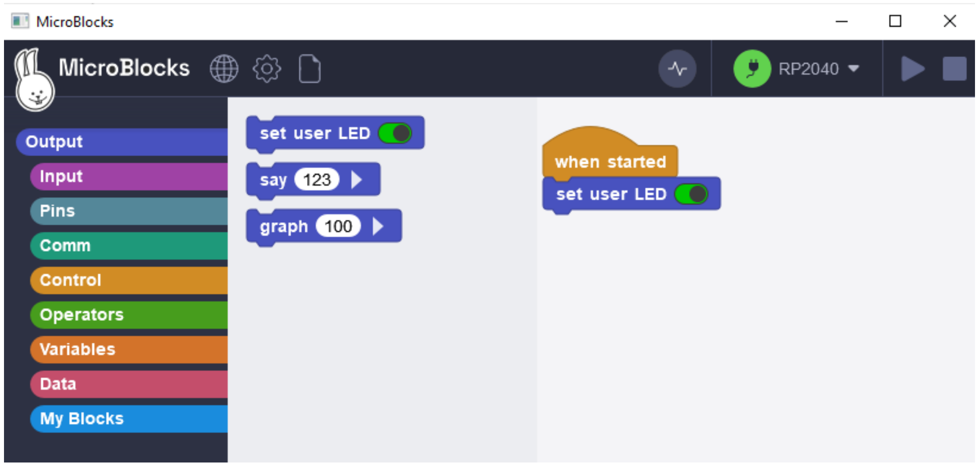
- Choose Raspberry Pi Pico
- The firmware will automatically copy to the Pico
- The Pico will reboot after installation and connects the Microblocks with PICO. Firmware installation is required only once



- In MicroBlocks, click Connect Select the USB device / COM port for Raspberry Pi Pico if it not automatically connected
- Wait for the status to show Connected.
- The Pico's onboard LED may blink, indicating a successful connection.

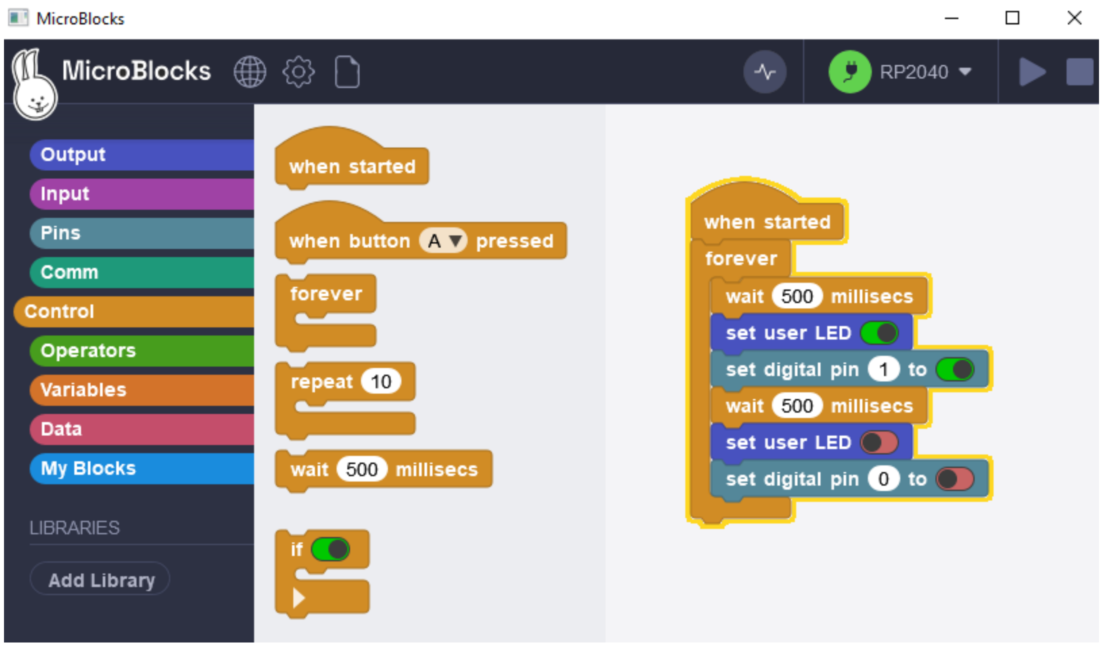
Step 3: Test the Connection by Write Your First Program - LED on

- Drag the "when started" block from the control palette to scripting area
- Drag the "Set user LED" block from the output palette to scripting area
- Then click on play button , The LED on the PICO board near to LED will glow



Step 4: Create a simple LED & pin Toggle program.

- Drag the “when started” block from the control palette to scripting area
- Drag the “Forever” block from the control palette to scripting area
- Inside the “Forever” block keep the LED & GPIO-1 pin and delay block as given in the below image
- Then RUN the program, you can see LED blinking with 500ms On & OFF pattern.



```
when started
forever
    wait 500 millisecs
    set user LED
    set digital pin 1 to
    wait 500 millisecs
    set user LED
    set digital pin 0 to
```

7. Getting Started with Thonny

7.1. What You'll Need

- Raspberry Pi Pico or Pico W
- Micro USB cable

- Computer (Windows, Mac, or Linux)
- Thonny IDE (free software)

7.2. Connect to Thonny

Step 1: Install Thonny IDE

- **Go to https://thonny.org**
- **Download the version for your operating system**
- **Run the installer and follow the installation prompts**
- **Launch Thonny after installation**

Note: Thonny comes with Python built-in, so you don't need to install Python separately.

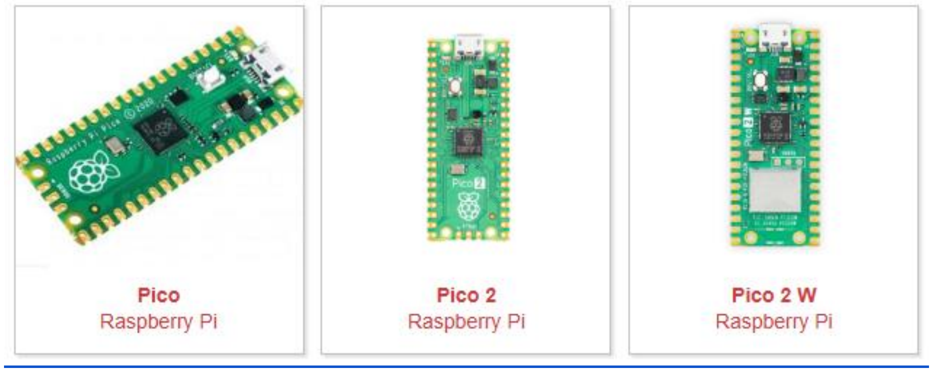
Step 2: Installing MicroPython Firmware:

Method 1: Using Thonny (Easiest)

- Connect your Pico to your computer using the micro USB cable **while holding down the BOOTSEL button** (the white button on the board)
- Keep holding BOOTSEL until the Pico appears as a USB drive named "RPI-RP2"
- Open Thonny
- Go to **Tools** → **Options** → **Interpreter**
- Select **"MicroPython (Raspberry Pi Pico)"** from the dropdown
- Click **"Install or update MicroPython"**
- Select your Pico from the device list
- Click **Install**
- Wait for the installation to complete

Method 2: Manual Installation

- Download the latest & appropriate MicroPython UF2 file from MicroPython - Python for microcontrollers



- Hold the BOOTSEL button while connecting your Pico via USB
- Drag and drop the UF2 file onto the RPI-RP2 drive
- The Pico will automatically reboot with MicroPython installed

Step 3: Configure Thonny for Pico

- Open Thonny
- Go to **Tools** → **Options** (or **Thonny** → **Preferences** on Mac)
- Click the **Interpreter** tab
- Select **"MicroPython (Raspberry Pi Pico)"** from the interpreter dropdown
- Select the correct port (usually auto-detected as "Board in FS mode")
- Click **OK**

You should now see the MicroPython shell at the bottom of Thonny showing something like:

```
MicroPython v1.xx on 2024-xx-xx; Raspberry Pi Pico with RP2040 >>>
```

Step 4: Write Your First Program - Blink LED

The Raspberry Pi Pico has a built-in LED connected to GPIO pin 25 (pin "LED" on Pico W).

Code:

python
```
from machine import Pin
import time

# Initialize the onboard LED
led = Pin(25, Pin.OUT)  # Use Pin("LED") for Pico W

# Blink loop
while True:
    led.on()          # Turn LED on
    time.sleep(0.5)   # Wait 0.5 seconds
    led.off()         # Turn LED off
 time.sleep(0.5)   # Wait 0.5 seconds
```

Running the Code:

1. Type or paste the code into Thonny's editor (top panel)
2. Click the **green Run button** (▶) or press **F5**
3. If prompted, save the file to your Pico as **main.py** (this makes it run on startup) or **blink.py**
4. Watch your Pico's LED blink!

To stop the program: Click the **Stop button** (■) or press **Ctrl+C**

Step 5: Understanding the Thonny Interface

Main Components:

- **Editor (Top):** Where you write your code
- **Shell (Bottom):** Interactive MicroPython REPL (Read-Eval-Print Loop)
- **Files Panel (Left):** Shows files on your computer and Pico
    - Enable it via **View → Files**

Working with Files:

Save to Pico:

1. Click **File → Save as**
2. Select **"Raspberry Pi Pico"**
3. Name your file (e.g., `main.py`)

Save to Computer:

1. Click **File → Save as**
2. Select **"This computer"**
3. Choose location and name

Step 6: Try Interactive Mode (REPL)

You can test code directly in the Shell without writing a program:

1. Click in the Shell area at the bottom
2. Type commands after the `>>>` prompt:

python
```
>>> from machine import Pin
>>> led = Pin(25, Pin.OUT)
>>> led.on()
>>> led.off()
```

This is great for testing individual components!

7.3. Common Pico Pin Functions

Basic GPIO Control:

python
```
from machine import Pin

# Output (LED, relay, etc.)
pin = Pin(15, Pin.OUT)
pin.on()
pin.off()
pin.toggle()

# Input (button, sensor, etc.)
button = Pin(14, Pin.IN, Pin.PULL_UP)
state = button.value()  # Returns 0 or 1
```

PWM (Brightness Control):

python

```
from machine import Pin, PWM

pwm = PWM(Pin(15))
pwm.freq(1000)       # Set frequency
pwm.duty_u16(32768)  # 50% brightness (0-65535)
```

7.4. Troubleshooting Tips

Pico not detected:

- Try a different USB cable (some are power-only)
- Hold BOOTSEL button when connecting
- Try a different USB port

Code not running:

- Check the Shell for error messages
- Make sure you selected the correct interpreter
- Try disconnecting and reconnecting the Pico

Permission errors (Linux):

bash

```
sudo usermod -a -G dialout $USER
```

Then log out and back in.

"Device is busy" error:

- Stop any running programs first
- Disconnect and reconnect the Pico

7.5. Next Steps

Now that you have Thonny set up, you can:

- Connect external LEDs, buttons, and sensors

- Use PWM for motor control
- Read analog sensors with ADC
- Communicate via I2C, SPI, or UART
- Build IoT projects with Pico W's WiFi

Resources:

- Official Documentation: https://www.raspberrypi.com/documentation/microcontrollers/
- MicroPython Guide: https://docs.micropython.org/
- Thonny Help: Help menu in Thonny
  https://www.waveshare.com/wiki/Raspberry_Pi_Pico
  https://thonny.org/