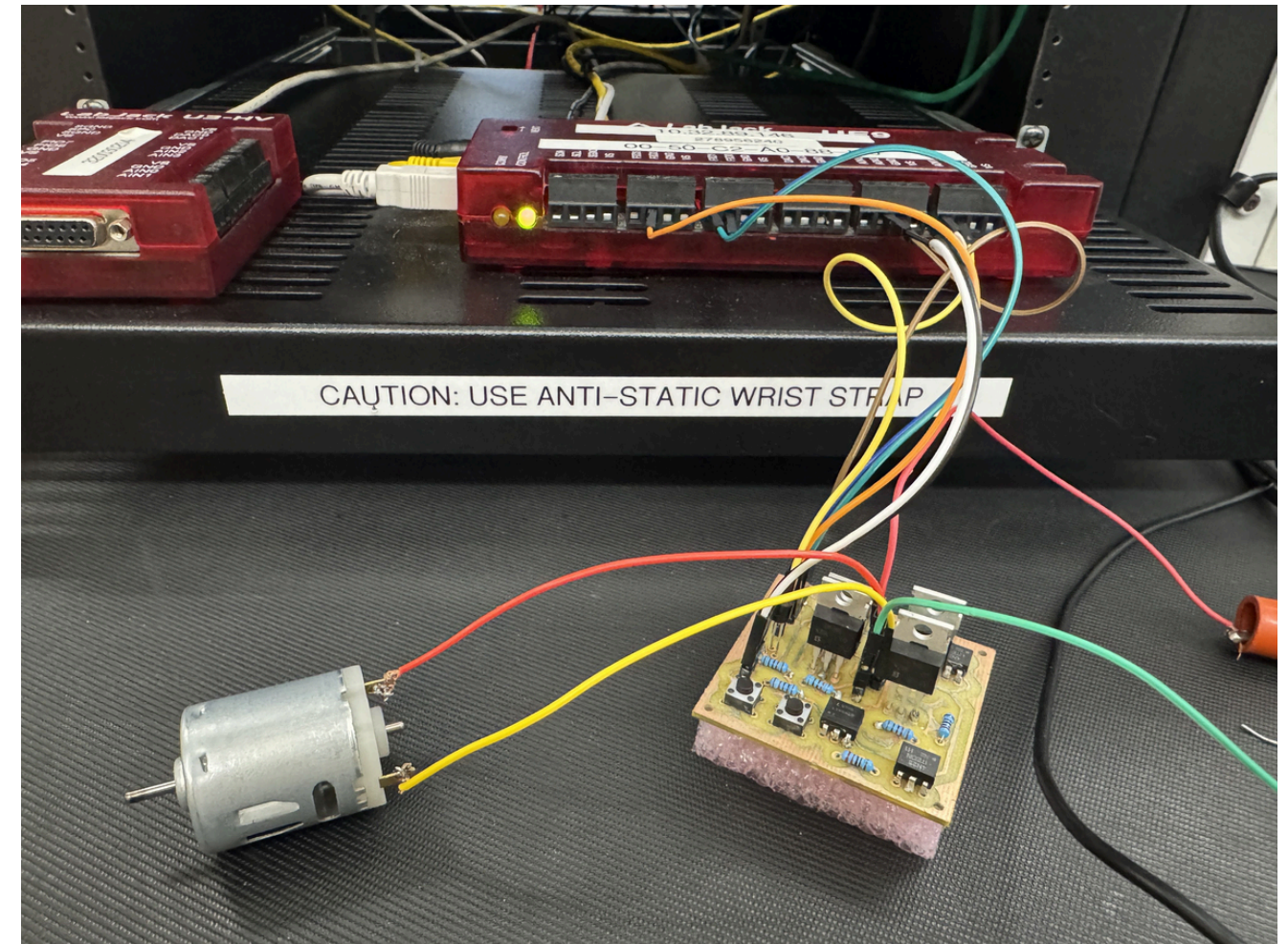


Motor Control and Emergency System Using LabJack UE9 and Python

Project Team

Team Members

- Lovepreet Singh (148872229)
- Saket Chahal (155140221)
- Hemant Vohra (149116220)
- Ankur Kashyap (149978223)



Abstract



Real-time DC Motor Control System

This project presents a real-time DC motor control system developed using the LabJack UE9 data acquisition module and Python.



Safety Implementation

The system includes **safety features** such as an **Emergency Stop** and **Reset** button to handle fault conditions.



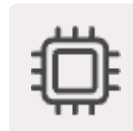
Software Architecture

Python's **multithreading** enables non-blocking user control and real-time terminal feedback.



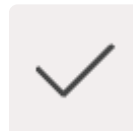
User Control Features

The goal was to build a system that allows users to control motor **speed** and **direction** through a software-based PWM signal.



Hardware Design

A **custom PCB** was designed using MOSFETs and optoisolators for safe and efficient control.



Validation

Tests confirmed accurate control, reliable emergency handling, and ease of use.

Introduction



Core Technology

Motor control is a core aspect of automation and robotics.



Safety Requirements

In critical applications, safety mechanisms like **emergency shutdowns** and **feedback monitoring** are essential.



Integrated Components

This project integrates:

- LabJack UE9 for analog/digital I/O
- Python interface for logic and display
- Emergency Stop and Reset buttons
- Temperature monitoring (TMP36 sensor)
- A custom PCB designed and fabricated by the team



Future Expansion

Designed to be **expandable** for future features like GUI and wireless control.

Problem Statement &

Problem Statement

Develop a system to safely control a DC motor's speed and direction using LabJack UE9 and Python with built-in emergency handling and real-time monitoring.

Project Goals

- Speed and direction control using PWM
- Emergency shutdown via AIN1
- Reset mechanism via AIN2 (hold for 2s)
- Live temperature monitoring (AIN0)
- Real-time terminal UI using Python
- Custom-built, cost-efficient PCB

System Overview

LabJack UE9 Interface

Controls digital outputs and reads analog sensor data. Interfaces with emergency, reset, and temperature sensors.

Sensors & Switches

TMP36: measures ambient temperature. Pushbuttons: Emergency Stop & Reset



Python Software (Multithreaded)

Accepts user input, sends PWM signals, monitors sensors, displays real-time system status

Custom PCB

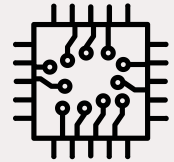
Contains optoisolators, MOSFETs, connectors. Provides electrical isolation and motor power handling

Circuit Schematic



Full System

Showing LabJack UE9 connected to analog sensors, digital outputs, and motor control components



Component

Emergency and reset button circuit, temperature sensor , MOSFETs controlling motor power



Safety

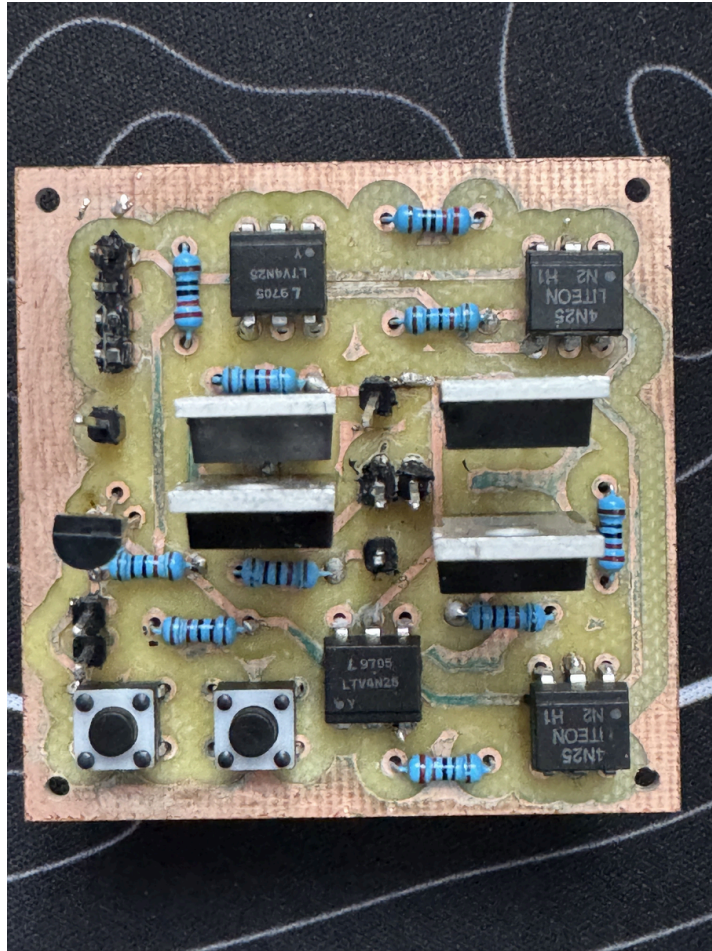
Optical isolation used for protecting LabJack from high-current spikes



Validation

Schematic created and tested using breadboard before PCB layout

PCB Design



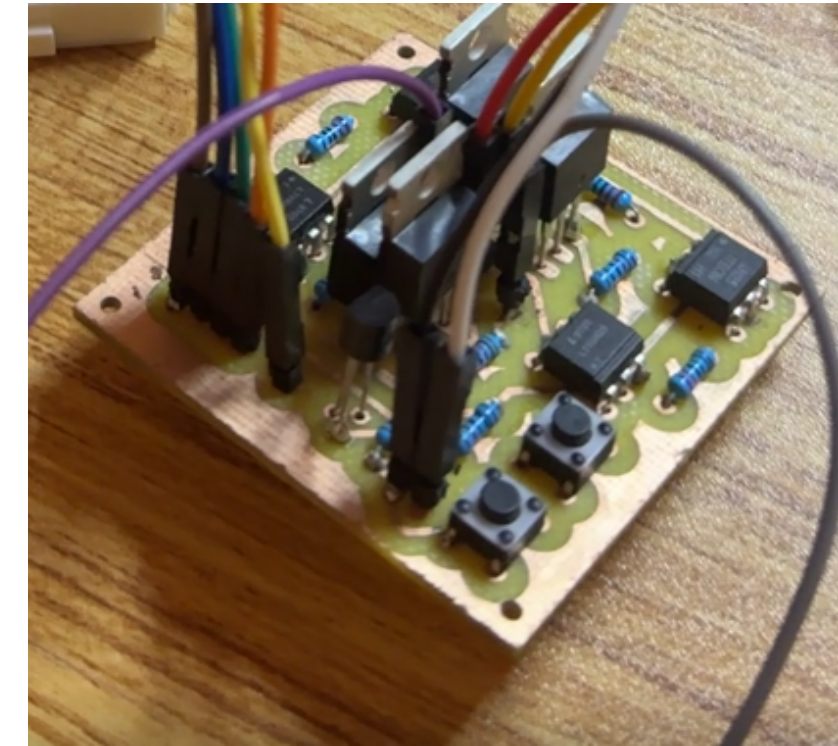
Top View

Clear trace routing to reduce noise and cross-talk. Component silkscreens labeled for easy assembly



3D Render

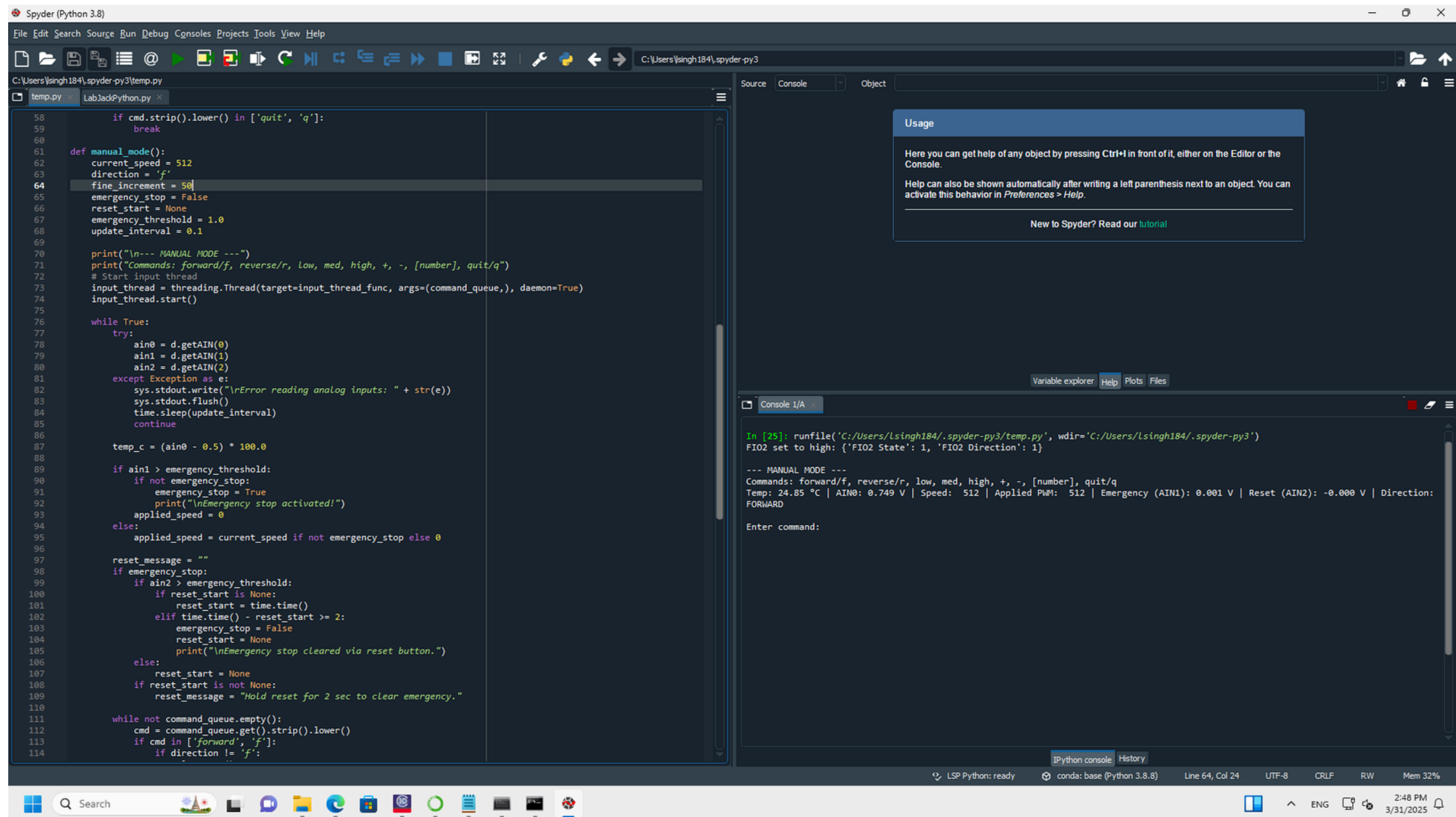
Visual of all mounted components: optos, MOSFETs, connectors. Final board simulated before printing



Completed

PCB printed and soldered manually. Components sourced with cost-efficiency in mind

Code / Console



The image shows the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.8)" and contains a code editor on the left and a console window on the right.

Code Editor: The code editor displays a Python script named "temp.py" with the following content:

```
58     if cmd.strip().lower() in ['quit', 'q']:
59         break
60
61     def manual_mode():
62         current_speed = 512
63         direction = 'f'
64         fine_increment = 50
65         emergency_stop = False
66         reset_start = None
67         emergency_threshold = 1.0
68         update_interval = 0.1
69
70         print("\n--- MANUAL MODE ---")
71         print("Commands: forward/f, reverse/r, low, med, high, +, -, [number], quit/q")
72         # Start input thread
73         input_thread = threading.Thread(target=input_thread_func, args=(command_queue,), daemon=True)
74         input_thread.start()
75
76         while True:
77             try:
78                 ain0 = d.getAIN(0)
79                 ain1 = d.getAIN(1)
80                 ain2 = d.getAIN(2)
81             except Exception as e:
82                 sys.stdout.write("\rError reading analog inputs: " + str(e))
83                 sys.stdout.flush()
84                 time.sleep(update_interval)
85                 continue
86
87             temp_c = (ain0 - 0.5) * 100.0
88
89             if ain1 > emergency_threshold:
90                 if not emergency_stop:
91                     emergency_stop = True
92                     print("\nEmergency stop activated!")
93                     applied_speed = 0
94             else:
95                 applied_speed = current_speed if not emergency_stop else 0
96
97             reset_message = ""
98             if emergency_stop:
99                 if ain2 > emergency_threshold:
100                     if reset_start is None:
101                         reset_start = time.time()
102                     elif time.time() - reset_start >= 2:
103                         emergency_stop = False
104                         reset_start = None
105                         print("\nEmergency stop cleared via reset button.")
106                 else:
107                     reset_start = None
108                 if reset_start is not None:
109                     reset_message = "Hold reset for 2 sec to clear emergency."
110
111             while not command_queue.empty():
112                 cmd = command_queue.get().strip().lower()
113                 if cmd in ['forward', 'f']:
114                     if direction != 'f':
```

Console Window: The console window shows the output of the script execution:

```
In [25]: runfile('C:/Users/Lsingh184/.spyder-py3/temp.py', wdir='C:/Users/Lsingh184/.spyder-py3')
FI02 set to high: {'FI02 State': 1, 'FI02 Direction': 1}

--- MANUAL MODE ---
Commands: forward/f, reverse/r, low, med, high, +, -, [number], quit/q
Temp: 24.85 °C | AIN0: 0.749 V | Speed: 512 | Applied PWM: 512 | Emergency (AIN1): 0.001 V | Reset (AIN2): -0.000 V | Direction: FORWARD

Enter command:
```

The console window also includes a "Usage" section with the following text:

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our [tutorial](#)

Python Terminal

User Interface

Simple, responsive interface built in Python using multithreading

Command Support

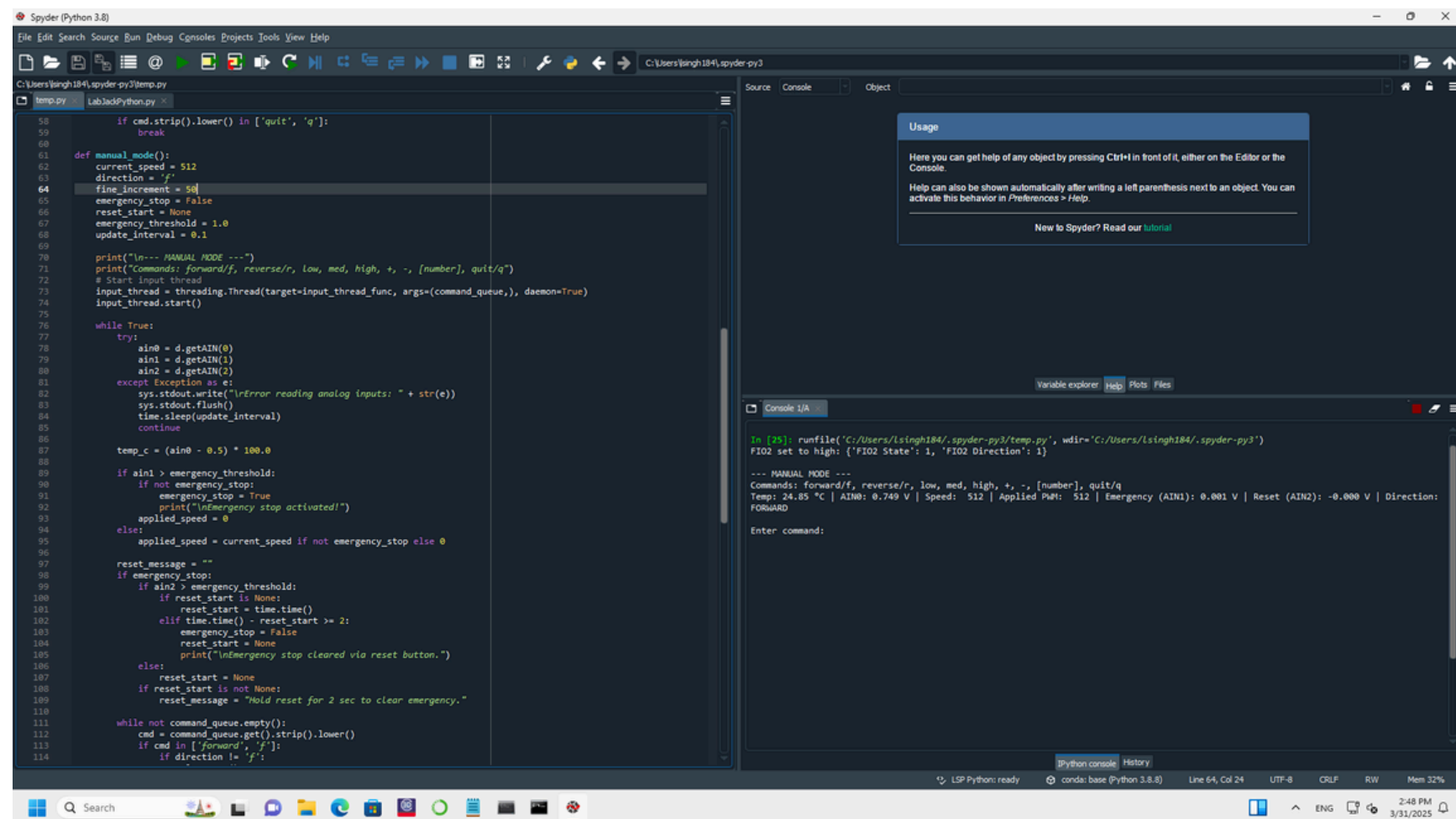
forward/f, reverse/r, speed presets, manual entry (0-1023), quit/q

Real-time Display

Shows current speed, temperature, emergency status, reset confirmation

Compatibility

No GUI required; can run on any basic terminal



```
50 if cmd.strip().lower() in ['quit', 'q']:
51     break
52
53 def manual_mode():
54     current_speed = 512
55     direction = 'f'
56     fine_increment = 50
57     emergency_stop = False
58     reset_start = None
59     emergency_threshold = 1.0
60     update_interval = 0.1
61
62     print("\n--- MANUAL MODE ---")
63     print("Commands: forward/f, reverse/r, low, med, high, +, -, [number], quit/q")
64     # Start input thread
65     input_thread = threading.Thread(target=input_thread_func, args=(command_queue,), daemon=True)
66     input_thread.start()
67
68     while True:
69         try:
70             ain0 = d.getAIN(0)
71             ain1 = d.getAIN(1)
72             ain2 = d.getAIN(2)
73         except Exception as e:
74             sys.stdout.write("\nError reading analog (inputs: " + str(e))
75             sys.stdout.flush()
76             time.sleep(update_interval)
77             continue
78
79         temp_c = (ain0 - 0.5) * 100.0
80
81         if ain1 > emergency_threshold:
82             if not emergency_stop:
83                 emergency_stop = True
84                 print("\nEmergency stop activated!")
85                 applied_speed = 0
86             else:
87                 applied_speed = current_speed if not emergency_stop else 0
88
89         reset_message = ""
90         if emergency_stop:
91             if ain2 > emergency_threshold:
92                 if reset_start is None:
93                     reset_start = time.time()
94                 elif time.time() - reset_start >= 2:
95                     emergency_stop = False
96                     reset_start = None
97                     print("\nEmergency stop cleared via reset button.")
98             else:
99                 reset_start = None
100             if reset_start is not None:
101                 reset_message = "Hold reset for 2 sec to clear emergency."
102
103         while not command_queue.empty():
104             cmd = command_queue.get().strip().lower()
105             if cmd in ['forward', 'f']:
106                 if direction != 'f':
```

```
In [25]: runfile('C:/Users/Lsingh184/spyder-py3/temp.py', wdir='C:/Users/Lsingh184/spyder-py3')
FIO2 set to high: {'FIO2 State': 1, 'FIO2 Direction': 1}
--- MANUAL MODE ---
Commands: forward/f, reverse/r, low, med, high, +, -, [number], quit/q
Temp: 24.85 °C | AIN0: 0.749 V | Speed: 512 | Applied PWM: 512 | Emergency (AIN1): 0.001 V | Reset (AIN2): -0.000 V | Direction: FORWARD
Enter command:
```

Results & Testing

Test Type	Result
Speed Control	0–1023 worked accurately
Direction Reversal	Safe reversal with 0.8s delay
Emergency Button	Immediate PWM stop
Reset Button	Restored function after 2s hold
Temperature Monitoring	Real-time updates on terminal

Performance Summary:

- Stable operation under long runtime
- Responsive to user input even during active loops
- All features worked in lab demonstration (A4070)

Team Member Contribution

Lovepreet Singh (148872229)

- Circuit testing and breadboard prototyping
- Assisted in final PCB assembly and soldering
- Emergency and reset switch circuit validation

Saket Chahal (155140221)

- Python code development with multithreading
- Integrated terminal interface with LabJack UE9
- Document formatting and project report compilation

Hemant Vohra (149116220)

- Designed PCB layout and component placement
- Managed BOM, sourced components
- Led lab testing and safety mechanism verification

Ankur Kashyap (149978223)

- Prepared schematics and 3D PCB rendering
- Coordinated team meetings and task allocation
- Final presentation creation and dry-run lead

References

Anaconda. (2024, October 14). Download Anaconda Distribution | Anaconda. <https://www.anaconda.com/download>

Digi-Key Electronics. (n.d.). Digi-Key Electronics – Electronic Components Distributor. <https://www.digikey.ca/?msockid=2f1c1869837a652f17000d268241647d>

LabJack. (n.d.). Data Acquisition Systems | LabJack DAQ Systems Analog & Digital I/O. <https://labjack.com/>