**📘 HSI Machine User Manual & Setup Guide**

**📂 Table of Contents**

1. System Requirements
2. Folder Structure
3. Spinnaker SDK Installation
4. Initial Setup
5. Virtual Environment Setup
6. Installing Required Libraries
7. Running the Code
8. Data Acquisition Workflow
9. Image Processing Modules
10. Troubleshooting
11. Contacts / Support

**📘 1. Project Introduction**

This Tabletop Hyperspectral Imaging (HSI) Machine is a lab-scale imaging platform designed to enable rapid acquisition and analysis of spectral data across visible and near-infrared wavelengths. Built using a FLIR camera, programmable motor stage, and Python-based image processing pipeline, this system is ideal for academic and industrial research in areas such as:

* Agriculture and crop analysis
* Biomedical imaging
* Material and surface inspection
* Food quality and moisture detection

The machine offers modularity, automation, and full Python integration for end-to-end data handling.

**2. System Requirements**

* **OS**: Windows 10 or later (64-bit)
* **Python**: Version 3.10 recommended
* **RAM**: 8 GB or higher
* **Storage**: SSD (optional)
* **Camera SDK**: Teledyne FLIR Spinnaker SDK
* **Tools**: Python, Git 1(optional), ImageJ (optional)

**📁 3. Folder Structure**

Access the Github Page:

<https://github.com/rahmani143/Intern-KFUPM>

root/

├── annotated\_lines.png

├── blue\_lines\_mask.png

├── blue\_vertical\_lines.png

├── calibration\_curve.png

├── debug\_detected\_lines.png

├── debug\_edges.png

├── debug\_mask.png

├── flow chart.png

├── aquired\_location.py

├── aquiring.py

├── cube\_synthesis.py

├── final.py

├── test\_intensity\_pic.py

├── test\_position\_pixel.py

1\_aquire\_video\_photo/

├── aquire\_graphical.py

├── aquire\_video.py

├── aquired\_location.py

├── arduino\_com\_detect.py

├── blue\_spectral\_recognize.py

├── main\_arduino\_graphical.py (important code)

2\_radiometric\_calibration/

├── radiometric\_4.py

├── radiometric\_calibration\_3.py

├── radiometric\_calibration\_single.py

├── radiometric\_generatemean\_pics\_1.py

├── radiometric\_white\_test\_2.py

3\_spatial\_calibraion/

├── 1\_spatial\_calibration\_1.py

├── 2\_spatial\_calibration\_json\_init\_2.py

├── 3\_spatial\_calibration\_test\_3.py

├── give\_axis.py

├── line\_extension\_edge\_detection.py

├── n-2\_find\_max\_intensity\_blue\_lines.py

├── notused\_edge\_detection.py

├── notused\_edge\_detection\_pointer.py

├── notused\_straighten\_image.py

├── plot\_curves.vertical\_alltogether.py

├── plot\_function\_calibration\_spatial.py

├── plot\_lines\_coordinates\_spectral.py

├── plot\_straight\_line\_one\_at\_a\_time.py

├── spatial\_hsi\_cube.py

4\_spectral\_calibration/

├── aggregate\_mean.py

├── detect\_blue\_line.py

├── find\_max\_intensity.py

├── mean.py

├── plot\_curves\_horizontal.py

├── spectral\_hsi\_cube.py

├── straighten\_image.py

5\_stage\_1\_Ardionousingpython\_final\_ready/

├── stage1\_arduino\_final.py

calibration\_total/

├── aquire\_video.py

├── band\_generation.py

├── calibration\_test.py

├── plot\_function\_calibration\_spectral.py

├── view\_cube.py

appending\_codes/

(This folder has the Arduino code in it (Do not touch))

**🔧 4. Spinnaker SDK Installation (FLIR)**

**🧭 Step-by-step Guide (Windows)**

Download Spinnaker SDK by Teledyne at :  
<https://www.teledynevisionsolutions.com/support/support-center/software-firmware-downloads/iis/spinnaker-sdk-download/spinnaker-sdk--download-files/?pn=Spinnaker+SDK&vn=Spinnaker+SDK>

( **disclaimer**: check your python version and install with respect to it

python --version )

**A. Add Spinnaker Paths to Environment Variables**

1. **Open System Environment Variables**
   * Press Win + R, type sysdm.cpl, press **Enter**
   * Go to the **Advanced** tab → Click **Environment Variables**
2. **Edit the Path Variable**  
   Under **System variables**, select Path → Click **Edit**  
   Add these two paths:

C:\Program Files\Teledyne\Spinnaker\bin64

C:\Program Files\Teledyne\Spinnaker\lib64

1. **Add PYTHONPATH Variable**
   * Click **New** under System Variables
     + Name: PYTHONPATH
     + Value:

C:\Program Files\Teledyne\Spinnaker\bin64;

C:\Program Files\Teledyne\Spinnaker\lib64

* + If PYTHONPATH already exists, click **Edit** and append both paths separated by “ ; “

1. **Apply Changes**  
   Click **OK** on all dialog boxes and **restart your terminal or IDE**

**( ⚠️ Disclaimer: Recommended Editor – Visual Studio Code (VS Code)**

This manual is designed with **Visual Studio Code (VS Code)** as the primary development environment. All paths, shortcuts, and terminal instructions are intended for use within VS Code’s integrated terminal on a Windows system.

**👉 Before You Begin:**

Please download and install **VS Code** from the official site:

🔗 <https://code.visualstudio.com/>

And Anaconda as well

🔗 <https://www.anaconda.com/download> )

**🧪 5. Initial Environment Setup**

**Create Conda Virtual Environment (Python 3.10)**

In cmd of the folder:

conda create -n pyspin\_env python=3.10 -y

conda activate pyspin\_env

**🔹 6. Install Required Python Libraries**

# 1. Activate the conda environment

conda activate pyspin\_env

# 2. Upgrade pip

python -m pip install --upgrade pip

# 3. Install the Spinnaker .whl package (adjust path if needed)

pip install "C:\Users\User\Downloads\spinnaker\_python-4.0.0.116-cp310-cp310-win\_amd64.whl"

# 4. Install all required Python libraries with the requirements.txt file

pip install -r requirements.txt

# 5. Optional: Install pyspin wrapper (community-maintained)

pip install pyspin # optional, if needed

**▶️ 7. Running the Code**

Activate the environment and run any Python script as follows:

conda activate pyspin\_env <python path\to\your\script.py>

Example:

Python flir\_test\1\_aquire\_video\_photo\aquired\_location.py

**📸 8. Data Acquisition Workflow**

Connect Camera via USB cable to Laptop

Ensure camera is detected by the FLIR Spinnaker SDK (via Device Manager)

Run GUI file :

1\_aquire\_video\_photo/main\_arduino\_graphical.py (File Location)

Launch GUI or use script to begin pushbroom scan:

python main\_arduino\_graphical.py

Captures ~200 images and stores in aquired\_images/raw/

**🧊 9. Image Processing Modules**

1. **Hypercube Creation**

Stacks all grayscale slices into shape (2048, 2048, 200)

python 2\_hypercube\_generation/hypercube.py

1. **Blue Line Detection via Regression**

python 3\_line\_detection/detect\_line.py

* Locates blue calibration line and extracts its trajectory

**🧠 10. Calibration Procedures**

Proper calibration is essential for accurate hyperspectral measurements. This section outlines three key types of calibration.

**A. Spatial Calibration**

* Capture calibration images using a target with vertical reference lines (e.g., blue lines on a white background).
* Apply edge detection using the Canny algorithm followed by line detection using Hough Transform.
* Measure pixel distances between detected vertical lines. Use known real-world distances between lines to compute a pixel-to-mm conversion factor.

**B. Rotation Correction**

* If lines appear slanted, use image preprocessing algorithms to straighten the image.
* Apply affine or perspective transforms based on line orientation to vertically align all reference lines.
* Ensure corrected images consistently reflect vertical calibration targets before proceeding to analysis.

**C. Spectral & Radiometric Calibration**

* Capture three sets of images:
  1. Dark Current Image – with lens cap on.
  2. White Reference Image – using a high-reflectance surface.
  3. Sample Image – actual scene or object.
* Apply the formula:

Reflectance = (Sample - Dark) / (White - Dark)

* Output calibrated reflectance spectra and save them in .csv or .json for further use.

**❗ 11. Troubleshooting**

| **Issue** | **Solution** |
| --- | --- |
| Spinnaker camera not found | Verify SDK installation & env variables |
| ModuleNotFoundError | Recheck pip install inside env |
| Image file errors | Check image path and formats |
| PYTHONPATH not effective | Restart CMD/IDE after setting it |
| DLL load failed errors | DLLs not in PATH → Check SDK path again |

|  |  |  |
| --- | --- | --- |
| **Issue** | **Cause** | **Suggested Fix** |
| **Slanted or rotated lines** | Camera alignment or improper mounting | Use rotation correction scripts with affine transform |
| **No vertical lines detected** | Low contrast or poor edge detection | Increase image brightness, tweak Canny thresholds |
| **NIR bands look noisy or misaligned** | Filter misplacement or out-of-focus image | Refocus lens and realign filter manually |
| **Spectral results inconsistent** | White/dark calibration issue | Ensure dark and white images are taken under same lighting |
| **Blue lines appear blurred** | Motion blur from motor stage | Reduce stage speed or stabilize platform |

**🧭 12. GUI User Interface Overview**

The graphical interface (main\_arduino\_graphical.py) offers real-time control and monitoring for image acquisition:

* 🔘 Start and stop scans using simple push-button commands.
* 📂 Automatic storage of captured images in the folder.
* 🔧 Adjustable parameters for scan length, speed, and resolution.
* ✅ Feedback window confirms success or errors during capture and saving.

Use the GUI for both manual operation and batch scanning.

**🧪 13. Testing & Validation**

Before beginning any experimental imaging, run the following checklist to validate the system:

* ✅ All components (camera, drivers, motors) are connected and functional.
* ✅ The system is spatially and spectrally calibrated.
* ✅ Test scans show uniform lighting and straight alignment.
* ✅ Known reference samples yield correct spectral response.
* ✅ All required software packages are installed and Python scripts run without errors.
* ✅ Calibration results (curves, conversion factors) are saved for reproducibility.

**📞 14. Contact / Support**

* **Email**: bss10i19ibrahimazeem@gmail.com
* **Organization**: National Institute of Technology, Warangal
* **Last Updated**: July 2025