SPROM:
Project Description Sheet

Prof. Dr. Natividad Martínez Madrid

Team: Elisa Veloso and Adam Roque

Summer Semester/25
Reutlingen

Initial Description

Project Title

Smart Proximity for Obstacle Monitoring

Acronym

SPROM

Short Description

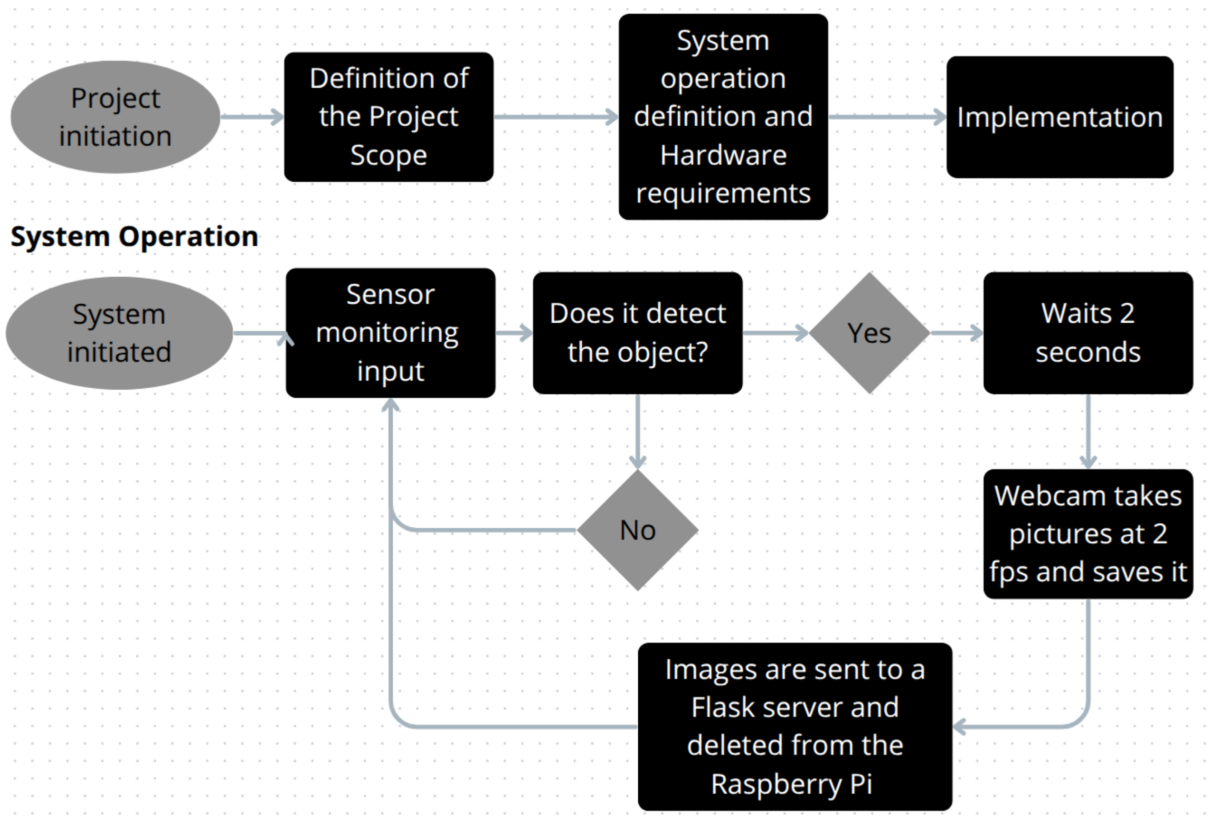
SPROM is a compact ultrasonic system built around a Raspberry Pi sensor that detects approaching objects, captures images at specific moments, and displays them on a local screen. Designed for educational and prototyping purposes, it provides an accessible and affordable alternative to complex object recognition systems..

Project Vision

Background and Motivation

In precision agriculture, it is important to monitor the collection of sugarcane from the fields. During this process, a truck harvests the sugarcane through a pipe and, upon arriving at the processing facility, a probe collects samples and deposits them into a bucket. Currently, trained machine learning models are used to detect the bucket's approach for sample collection. However, maintaining these models is complex and highly specific to each installation. By implementing a system that detects the bucket's approach without relying on heavy model training, we can generalize the process, reduce maintenance efforts, and make the operation more fluid, accurate, and efficient.

System Overview


Project initiation
Definition of the Project Scope
System operation definition and Hardware requirements
Implementation
System Operation
System initiated
Sensor monitoring input
Does it detect the object?
Yes
Waits 2 seconds
Webcam takes pictures at 2 fps and saves it
Images are sent to a Flask server and deleted from the Raspberry Pi
No

Goal

Intended Purpose (Zweckbestimmung)

The SPROM (Smart Proximity for Obstacle Monitoring) system is designed to detect the approach of objects in agricultural contexts, particularly in environments where physical sampling or interaction with moving components is required. Its primary purpose is to serve as a reliable, low-maintenance, and affordable alternative to complex object detection systems, especially in use cases such as sugarcane sample collection in precision agriculture. SPROM aims to increase operational efficiency and precision while reducing dependency on computationally intensive and installation-specific machine learning models.

Intended Use (bestimmungsgemäßer Gebrauch)

SPROM is intended to be installed in fixed positions near zones where object detection is critical, such as near sampling stations or industrial input points. The system consists of an ultrasonic sensor connected to a Raspberry Pi, which remains in standby mode until it detects an approaching object. Upon detection, the system captures an image of the event and displays it locally on a screen or via a web interface. Typical use includes configuring the device through a simple setup interface, positioning the sensor for optimal coverage of the target area, and using the visual feedback to monitor the process in real time. It is particularly suited for prototyping, research, and educational applications where ease of use and adaptability are key.

Features and Performance

Functional Requirements (Features)

1. **Object Proximity Detection:** Detects approaching objects using an ultrasonic sensor with a configurable detection range.

2. **Image Capture on Trigger:** Automatically captures an image when an object is detected within a predefined distance.

3. **Remote Visualization:** Displays captured images and detection status via a web interface.

4. **Event Logging:** Records detection events and timestamps for monitoring and analysis.

5. **System Initialization and Standby Modes:** Operates in predefined stages, including initialization, monitoring, and trigger response.

Non-Functional Requirements (Performance):

1. **Low Latency Detection**: The system must respond to object detection within 1 second.

2. **High Availability**: The system should operate continuously with minimal downtime in typical environmental conditions.

3. **Low Maintenance**: Designed to operate with minimal calibration or retraining, unlike traditional machine learning-based systems.

4. **Affordability**: Utilizes cost-effective hardware (Raspberry Pi and ultrasonic module) to maintain low production costs.

5. **Ease of Deployment**: System setup should take less than 30 minutes, including mounting, calibration, and configuration.

6. **Modularity**: The system architecture should allow for easy replacement or upgrading of components such as the ultrasonic sensor or display module.

Exclusion Criteria:

Despite correct installation and use, there are specific conditions under which the SPROM system may not fulfill its intended purpose. These exclusion criteria define the limitations of the system:

1. **Extreme Environmental Conditions**: The system is not designed for use in environments with excessive dust, moisture, heat, or electromagnetic interference, which may affect ultrasonic sensor accuracy and Raspberry Pi stability.

2. **High-Speed Object Movement**: Objects moving at speeds beyond the ultrasonic sensor's detection capabilities may not be accurately detected or photographed in time.

3. **Obstructed Detection Zone**: If the sensor's field of view is physically blocked or the object does not enter the designated detection zone, detection and image capture will not occur.

4. **Reflective or Non-Reflective Surfaces**: Some materials may interfere with ultrasonic signal reflection, such as highly absorbent, transparent, or irregularly shaped surfaces, reducing detection reliability.

5. **Power Supply Instability**: Inconsistent or insufficient power supply can cause system reboots or failure to operate, particularly during image processing or display.

6. **Incorrect Mounting or Alignment**: Improper installation angle or placement may cause the sensor to miss the object or misfire triggers.

7. **Software Modification**: Unauthorized or incorrect changes to the software or configuration may lead to malfunction or loss of core functionality.

Project Implementation

Hardware Requirements

Components

- Ultrasonic distance sensor, HC-SR04;
- Raspberry Pi Zero 2W, 4x 1GHz, 512MB RAM, WiFi, BT;
- Arducam 5MP OV5647 Miniature Camera Module
- Wires and breadboard for the circuits
- 1 kΩ and 2 kΩ resistors
- Metallic structure to hold the components.

- Bucket and rope to simulate the Operation on the Sugar Cane Mills

Demonstration Infrastructure

For the demonstration setup, a structure with a metalic will be used to support the proximity sensor, the Raspberry Pi, and the ArduCam. In front of this structure, the bucket will be placed. Using a rope, We will pull the platform to simulate the movement of the rail that supports the bucket in a sugar cane mill.

Software Requirements

- Language used: Python
- Framework: Flask (Html, css, Javascript)

Image Upload and Gallery System

To simulate real-time operation in sugarcane processing plants, the proximity detection system runs continuously on the Raspberry Pi. The ultrasonic sensor is constantly monitoring the area until it detects an object closer than 30 centimeters.

When such an object is detected, a trigger is sent to the camera module, which captures 6 consecutive images at 2 frames per second (fps) over a 3-second window.

This setup is intended to replicate how a bucket and material dumping would be detected in an actual industrial scenario, ensuring that relevant events are properly captured through images.

This project is also powered by "Flask", a lightweight framework for building web apps in Python. For image processing, we use "OpenCV", which helps us check if an image is sharp or blurry.

When someone uploads an image through the "/upload" route, the app saves it in a folder called "uploaded_images". If the folder doesn't exist yet, the app creates it automatically.

To keep things organized, each uploaded image is renamed with the "current date and time" for example: 2025-06-26_15-42-10_filename.jpg. This way, filenames are always unique and give a clear idea of when the image was added. Since the frame rate for this project is 2fps, the imagens from the same timestamp will be numbered as p1 or p2.

A function called "is_well_framed()" checks how sharp the image is. It uses the "Laplacian variance" to detect if the photo is blurry. If the image is sharp, it gets the prefix "OK_". If not, it gets "CROSS_". This makes it super easy to tell at a glance whether an image passed the quality check.

The homepage ("/") acts as a gallery. You can filter the images by date, time, or whether they're “OK” or “Not OK”. The app reads the info from the filenames and only shows the ones that match your filters. Under each image, there's a dropdown menu where it is possible to manually change the status. When you pick a new one, the app updates the filename in the background. It's a way to balance automation with human control.

There's also a route like "/images/<filename>", which sends the image to the browser, so it shows up on the page using regular "<img>" tags.

The function "is_well_framed(image_path)" is doing the actual sharpness check and returns "True" or "False" depending on the result.

The "weekly_cleanup()" function runs automatically every Sunday at midnight. It deletes everything in the upload folder and starts fresh.

The HTML Side

The front of the app is built with HTML and styled using CSS through Flask templates.

At the top of the page, there's a "filter bar". You can choose a specific day, a time range, and whether you want to see "OK" or "Not OK" images. When you click submit, the gallery updates with just the images you asked for.

Images appear in a "grid layout", and each one has a little caption showing its filename and status. There's also a dropdown menu under every image that lets you change its status. As soon as you make a selection, the form submits automatically, and the change is made in the backend.

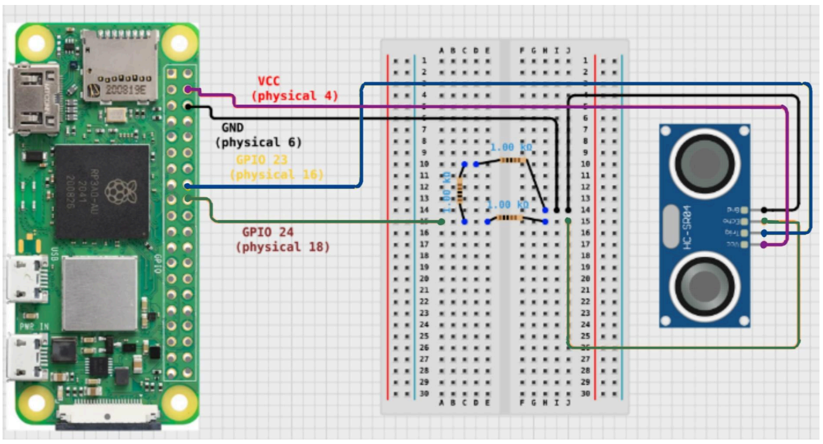
Source Code

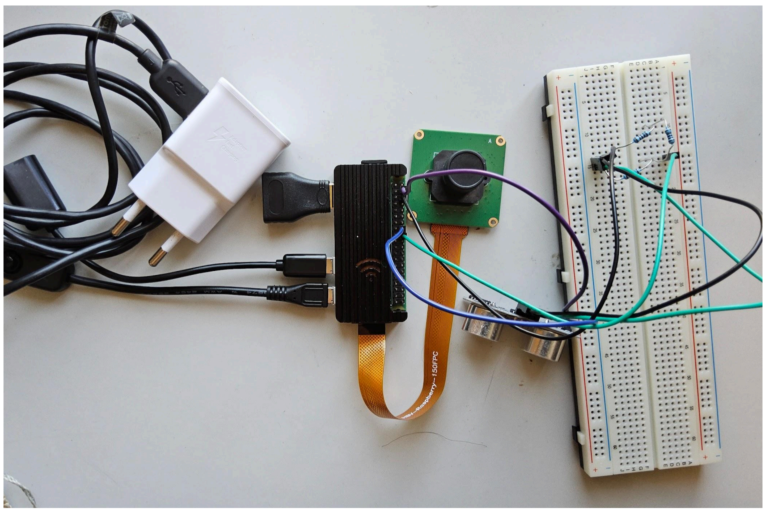
The source code was developed incrementally throughout the project. All scripts used, as well as different versions of the project description, were uploaded to GitHub and can be accessed at the following link: [link].

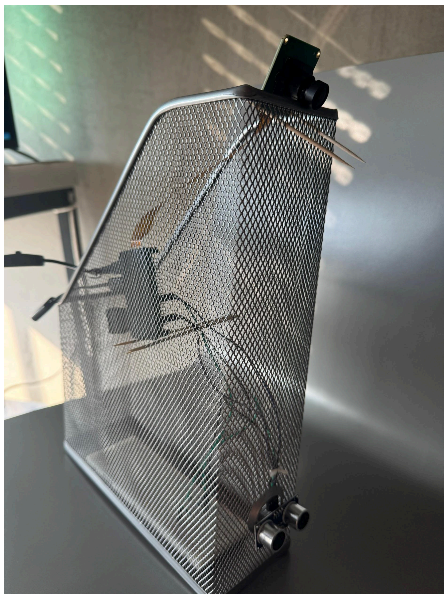
How to run this project

1. For running this project, a Raspberry Pi OS needs to be installed using the Raspberry Pi Imager. After setting up the Pi, the SSH needs to be enabled in the boot, as well as the Wifi.
2. With the Pi turned on, another client can be connected to it via SSH with the command: `ssh pi@<raspberry-pi-ip>`
3. After connecting to the Pi, the camera will be automatically enabled, and the system needs to be updated. To do this, run the command `sudo apt update && sudo apt full-upgrade -y`
4. Next, install the dependencies on the Raspberry Pi with `sudo apt install -y python3-picamera2 python3-flask python3-requests libcamera-apps`
5. Connect the Ultrasonic Sensor to the Raspberry Pi:
    a. VCC → 5V
    b. GND → GND

c. TRIG → GPIO 23 (pin 16)
d. ECHO → GPIO 24 (pin 18) → Here, use resistor divider to reduce 5V to 3.3V







6. In your Raspberry Pi home folder, place the script sensor_camera_uploader.py and update the line with your server's IP: `SERVER_URL = "http://<your-laptop-ip>:5000/upload"`
7. Create a systemd service /etc/systemd/system/sprom.service:

   ```
   [Unit]
   Description=SPROM sensor-camera uploader
   After=network.target

   [Service]
   ExecStart=/usr/bin/python3 /home/pi/sensor_camera_uploader.py
   Restart=always
   User=pi

   [Install]
   WantedBy=multi-user.target
   ```

   Then, enable it with:

   ```
   sudo systemctl daemon-reload
   sudo systemctl enable sprom
   sudo systemctl start sprom
   ```

8. Set Up the Flask Server on Your Laptop by navigating to your project directory and creating a virtual environment. Then install Flask with `pip install flask` and then run the server with `python app.py`. Finally open your browser on http://localhost:5000/

Conclusion

The SPROM system presents an effective alternative to complex machine learning-based object detection solutions. By combining a simple ultrasonic sensor, a Raspberry Pi, and a lightweight image-processing web application built with Flask and OpenCV, we developed a reliable, low-cost solution for monitoring physical events—such as sample collection in agricultural environments.

SPROM paves the way for accessible automation, especially in contexts where simplicity, speed, and accuracy are key.

Although we faced some challenges, especially as this was our first project involving hardware, we achieved successful results and significantly enhanced our understanding of IoT systems. This experience sparked further interest in the field and strengthened our motivation to continue developing our microcontroller and embedded systems skills.

References

Raspberry Pi Documentation. Available at: https://www.raspberrypi.com/documentation/

Cirkit Designer. Raspberry Pi Zero W-Based Ultrasonic Distance Measurement System. Available at: https://docs.cirkitdesigner.com/project/published/95485b93-f08f-4b06-ae18-4936a0e8382d/raspberry-pi-zero-w-based-ultrasonic-distance-measurement-system

Raspberry Pi. Getting Started with Your Raspberry Pi. Available at: https://www.raspberrypi.com/documentation/computers/getting-started.html#set-up-your-raspberry-pi

Amazon. For purchasing the hardware components.