
# Image Processing and Detection Algorithm

**Shonan Hendre**

**ECE 425- Microprocessor Systems**

**Professor Aaron Nanas**

## Introduction

The objective of this project is to design and implement a complete edge detection system on an embedded microcontroller (the Tiva™ TM4C123GH6PM) using UART communication and grayscale image processing techniques. This project integrates concepts from digital signal processing, embedded systems, and computer vision, demonstrating how resource-constrained hardware can perform fundamental image analysis tasks that are typically handled by more powerful systems.

## Background and Methodology

This project integrates several foundational embedded systems concepts:

1. Serial Communication (UART): Implemented low-level UART drivers to receive and transmit raw image data using memory-mapped I/O and register-level control.
2. Memory-Constrained Image Processing: Processed 64x64 grayscale images entirely in SRAM, demonstrating static memory management and optimized buffer usage without dynamic allocation.
3. Algorithm Implementation on Hardware: The Sobel edge detection algorithm — typically used in computer vision — was implemented in pure C using only integer arithmetic, optimized for real-time performance on a microcontroller.

The outcome is a fully functional edge detection system running on the TM4C123GH6PM, which receives a 64x64 grayscale image from a PC over UART and performs edge detection using the Sobel operator. It then sends the processed edge map back to the PC for visualization. The project demonstrates the feasibility of basic computer vision on a low-power embedded system with limited memory and no floating-point unit.

**Hardware & Software Components**

| Description | Quantity | Manufacturer |
|---|---|---|
| Tiva C Series TM4C123G LaunchPad | 1 | Texas Instruments |
| USB-A to Micro-USB Cable | 1 | N/A |
| EduBase Board | 1 | Trainer4Edu |

**Peripherals Used:**

SysTick (millisecond delays)

UART0 (serial data transfer)

GPIO Port A (PA0, PA1 for UART RX/TX)

**Software Libraries & Tools:**

- Keil µVision 5 IDE
- Python script to convert an image into 64x64 grayscale image png
- TeraTerm to log binary bytes of processed image from the Sobel algorithm
- MATLAB to reconstruct binary image from TeraTerm binary file

## Components Used

UART0 - Image input/output via serial communication

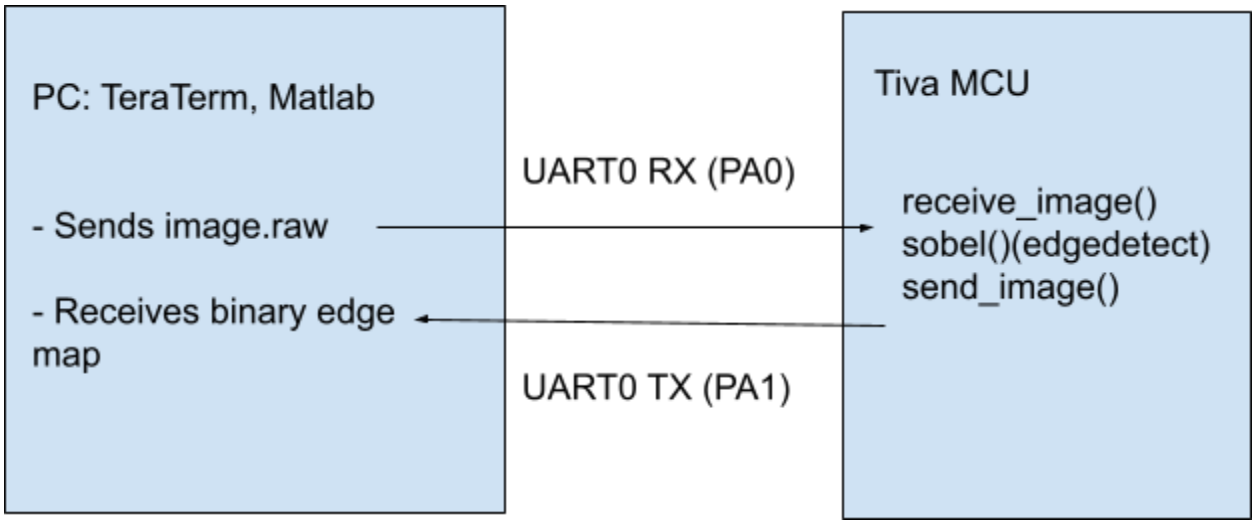
GPIO Port A - UART TX/RX pins (PA0, PA1)

SysTick- Timing performance/Generating Delays

**Pinout Used**

| Signal | PIN | PORT | Description |
|---|---|---|---|
| UART0 (RX) | PA0 | GPIOA, Pin 0 | UART receiver |
| UART0 (TX) | PA1 | GPIOA, Pin 1 | UART transmitter |

**Block Diagram**




## Analysis and Results

The project successfully executed the full image processing pipeline on the TM4C123GH6PM microcontroller. Below are the key results:

- A 64x64 grayscale image was sent from a PC to the microcontroller via UART0 using Tera Term.
- The MCU received and stored the image in SRAM using a blocking UART driver.
- The Sobel edge detection algorithm processed the image using integer math, computing horizontal and vertical gradients pixel-by-pixel.

- The resulting edge-detected image was sent back to the PC over UART and saved as teraterm.bin.
- The image was visualized using MATLAB using imshow() with proper orientation correction.

Video Link for Demonstration: https://vimeo.com/1082353849/5c87082c9f

## Challenges Encountered

### UART Transmission Issues

No bytes received initially due to:

1. Mismatched baud rate configuration (9600 baud expected, wrong divisor used).
2. Tera Term not set to "Binary" mode during file transfer/logging.

Fix: Calculated correct UART baud rate divisors and double-checked Tera Term settings.

### SRAM Limits

Trying to process larger images (e.g., 128×128) resulted in stack overflow or corrupted memory.
Fix: Settled on 64×64 size

## Conclusion

Through the development and implementation of this embedded edge detection system, I gained hands-on experience in applying core embedded systems concepts to solve a real-world signal processing problem under hardware constraints.
I learned how to configure and use UART communication at the register level to transmit binary image data between a microcontroller and a host PC and experienced the importance of memory management in embedded systems, especially when working with image buffers on a microcontroller with limited SRAM.
I deepened my understanding of the Sobel edge detection algorithm, including how it detects intensity gradients using convolution and how its performance can be approximated using integer

arithmetic to make it MCU-friendly. I explored the relationship between kernel design and edge sensitivity, and understood how 2D spatial filters operate at the pixel level. Finally I also learned how to combine hardware configuration, image processing, and data visualization to build a complete data pipeline: from image input → embedded processing → image output → PC visualization.