VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Winter Semester 2024-25

Laboratory Assignment – 2: Sensors &Actuators

Name-	Harsh Prakash
Reg. No.-	22BCT0098
Course Code-	BCSE311P
Prof. Name -	SIVANESAN S

Fundamentals of Optical Sensors - Module-3 Summary

1. Principles of Optical Sensing & Quantum Effects

Before implementing optical sensors, it is essential to understand:

- **Photoelectric Sensors**: Operate based on light absorption and electron emission.
- **Charge-Coupled Devices (CCD)**: Convert photons into electron signals for imaging.
- **Thermal Optical Sensors**: Detect infrared radiation for temperature measurement.
- **Active Infrared (AFIR) Sensors**: Utilize infrared beam interruption for motion and object detection.

---

2. Key Equations for Optical Sensors

Understanding sensor output, sensitivity, resolution, and Signal-to-Noise Ratio (SNR).

2.1 Photoelectric Sensor Equations

- **Photon-Based Photoelectric Current Equation:** I=q∗ϕ∗ηI = q * \phi * \eta where:
  - II = Photocurrent (A)
  - qq = Charge of an electron (1.6×10−19C1.6 \times 10^{-19} C)
  - ϕ\phi = Photon flux (photons/sec)
  - η\eta = Quantum efficiency (0 to 1)
- **Optical Power-Based Photoelectric Current Equation:** I=η∗Popt∗RI = \eta * P_{opt} * R where:
  - PoptP_{opt} = Incident optical power (W)
  - RR = Responsivity (A/W)
- **Sensitivity:** S=IPS = \frac{I}{P}
- **Resolution:** R=1SR = \frac{1}{S}
- **Signal-to-Noise Ratio (SNR):** SNR=IσnSNR = \frac{I}{\sigma_n}
  where σn\sigma_n = Noise current (A)

2.2 CCD Sensor Equations

- **Output Voltage of a CCD Pixel:** V=q∗NCV = \frac{q * N}{C} where:
  - NN = Number of photo-generated electrons

    - CC = Capacitance of the CCD pixel (F)
- **Quantum Efficiency (QE):** QE=NϕQE = \frac{N}{\phi}
- **Dynamic Range (DR):** DR=20log⁡10(NmaxNmin)DR = 20 \log_{10} \left(\frac{N_{max}}{N_{min}} \right) where NmaxN_{max} and NminN_{min} are the maximum and minimum electron counts.

Sensitivity of a CCD Sensor

- **Photon-Based CCD Sensitivity:** S=(η∗q)(h∗c)∗λS = \frac{(\eta * q)}{(h * c)} * \lambda where:
    - hh = Planck's constant (6.626×10−34Js6.626 \times 10^{-34} J s)
    - cc = Speed of light (3.0×108m/s3.0 \times 10^8 m/s)
    - λ\lambda = Wavelength of incident light (m)
- **Power-Based CCD Sensitivity:** S=ηRS = \frac{\eta}{R}
- **Voltage-Based Sensitivity:** S=VPS = \frac{V}{P}
- **Resolution of a CCD Sensor:** R=1SR = \frac{1}{S}

2.3 Active Infrared (AFIR) Sensor Equations

- **Infrared Sensor Power Output:** Pout=Pin∗e−αdP_{out} = P_{in} * e^{-\alpha d} where:
    - α\alpha = Absorption coefficient of the medium
    - dd = Distance traveled by the infrared beam
- **Object Detection Condition:** Pthreshold>PreceivedP_{threshold} > P_{received}
- **Sensitivity of an AFIR Sensor:** S=ΔPoutΔdS = \frac{\Delta P_{out}}{\Delta d}
- **Resolution of an AFIR Sensor:** R=1SR = \frac{1}{S}

This summary captures the essential equations and principles necessary for understanding optical sensors.

Overview

This C program simulates three distinct types of sensors:

1. **AFIR Sensor (Absorption-based Far Infrared Sensor)**
2. **CCD Sensor (Charge-Coupled Device Sensor)**
3. **Photoelectric Sensor**

Each sensor's functionality is implemented in separate header files:

- **afir_sensor.h** (Handles infrared absorption and object detection)
- **ccd_sensor.h** (Simulates imaging sensor characteristics)
- **photoelectric_sensor.h** (Models photoelectric effect for optical sensing)

The main program (sensor.c) integrates these components and runs simulations with sample values.

AFIR Sensor

The **Absorption-based Far Infrared (AFIR) Sensor** measures infrared radiation absorption over a distance and determines object presence based on received power.

Key Functions:

- compute_afir_power_output(): Computes how power diminishes over distance due to absorption.
- detect_object(): Determines if an object is detected based on a set power threshold.
- compute_afir_sensitivity(): Evaluates the sensor's response to power changes over distance.
- compute_afir_resolution(): Calculates resolution as the inverse of sensitivity.

CCD Sensor

A **Charge-Coupled Device (CCD) Sensor** is commonly used in imaging to convert incident photons into an electrical signal.

Key Functions:

- compute_ccd_output_voltage(): Computes the sensor's output voltage based on charge accumulation.
- compute_ccd_qe(): Determines quantum efficiency by analyzing electron-to-photon conversion.
- compute_ccd_dynamic_range(): Measures the range of detectable signals in decibels (dB).
- compute_ccd_sensitivity_photon(): Computes sensitivity considering photon energy (Planck's constant and light speed).

- compute_ccd_sensitivity_power(), compute_ccd_sensitivity_voltage(): Calculate sensitivity based on power and voltage.
- compute_ccd_resolution(): Derives resolution from sensitivity.

Photoelectric Sensor

The **Photoelectric Sensor** detects light intensity and converts it into an electrical current, useful in automation and object detection applications.

Key Functions:

- compute_photoelectric_current_flux(): Computes the photocurrent based on incoming photon flux and efficiency.
- compute_photoelectric_current_power(): Determines current output based on optical power.
- compute_sensitivity(): Calculates sensor sensitivity by relating photocurrent to optical power.
- compute_resolution(): Computes resolution from sensitivity.
- compute_snr(): Evaluates signal-to-noise ratio (SNR) in decibels, indicating the sensor's noise performance.

Main Program (sensor.c)

The main function initializes test parameters and invokes sensor functions, displaying computed values such as power, voltage, efficiency, and SNR. A randomization element (srand(time(NULL))) ensures variability in simulations.

Sensor Type	Primary Function	Core Computations
AFIR Sensor	Measures infrared absorption and detects objects	Power loss, object detection, sensitivity, resolution
CCD Sensor	Converts photons into electrical signals	Voltage output, quantum efficiency, dynamic range, sensitivity, resolution
Photoelectric Sensor	Detects light and converts it to current	Photocurrent, sensitivity, resolution, SNR

Conclusion

This modular implementation effectively models different sensor functionalities, allowing efficient testing of power variations, object detection thresholds, and response sensitivities. The program provides insights into sensor performance and can be extended for further applications in optical and infrared sensing.

CODE:

afir_sensor.h

```
#ifndef AFIR_SENSOR_H
#define AFIR_SENSOR_H
#include <math.h>
#define AFIR_THRESHOLD 0.5

double compute_afir_power_output(double initial_power, double distance) {
    return initial_power * exp(-0.1 * distance);
}

int detect_object(double power) {
    return power < AFIR_THRESHOLD ? 1 : 0;
}

double compute_afir_sensitivity(double power, double distance) {
    return fabs(compute_afir_power_output(power, distance) - power) / distance;
}

double compute_afir_resolution(double sensitivity) {
    return 1.0 / sensitivity;
}
#endif

/* ccd_sensor.h */
#ifndef CCD_SENSOR_H
#define CCD_SENSOR_H
#include <math.h>
#define PLANCK_CONSTANT 6.626e-34
#define LIGHT_SPEED 3.0e8
```

```
double compute_ccd_output_voltage(double charge, double capacitance) {
   return charge / capacitance;
}

double compute_ccd_qe(double incident_photons, double generated_electrons) {
   return generated_electrons / incident_photons;
}

double compute_ccd_dynamic_range(double max_signal, double noise_floor) {
   return 20 * log10(max_signal / noise_floor);
}

double compute_ccd_sensitivity_photon(double wavelength) {
   return PLANCK_CONSTANT * LIGHT_SPEED / wavelength;
}

double compute_ccd_resolution(double sensitivity) {
   return 1.0 / sensitivity;
}
#endif
```

photoelectric_sensor.h

```
#ifndef PHOTOELECTRIC_SENSOR_H
#define PHOTOELECTRIC_SENSOR_H
#include <math.h>

double compute_photoelectric_current_flux(double photon_flux, double efficiency) {
   return photon_flux * efficiency;
}
```

```
double compute_photoelectric_current_power(double optical_power, double efficiency) {
    return optical_power * efficiency;
}

double compute_sensitivity(double photocurrent, double optical_power) {
    return photocurrent / optical_power;
}

double compute_resolution(double sensitivity) {
    return 1.0 / sensitivity;
}

double compute_snr(double signal, double noise) {
    return 20 * log10(signal / noise);
}
#endif
```

sensor.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "afir_sensor.h"
#include "ccd_sensor.h"
#include "photoelectric_sensor.h"

int main() {
    srand(time(NULL));

    double afir_initial_power = 10.0;
    double afir_distance = 5.0;
    double afir_power_output = compute_afir_power_output(afir_initial_power, afir_distance);
    printf("AFIR Power Output: %lf\n", afir_power_output);
    printf("Object Detected: %s\n", detect_object(afir_power_output) ? "Yes" : "No");

    double charge = 5e-12, capacitance = 2e-12;
    printf("CCD Output Voltage: %lf V\n", compute_ccd_output_voltage(charge, capacitance));
    printf("CCD Quantum Efficiency: %lf\n", compute_ccd_qe(1000, 800));
    printf("CCD Dynamic Range: %lf dB\n", compute_ccd_dynamic_range(10000, 10));

    double optical_power = 0.005, efficiency = 0.85;
    printf("Photoelectric Current: %lf A\n", compute_photoelectric_current_power(optical_power,
efficiency));
    printf("Photoelectric Sensitivity: %lf\n", compute_sensitivity(0.002, optical_power));
    printf("Photoelectric SNR: %lf dB\n", compute_snr(5.0, 0.01));

    return 0;
}
```