Problem Statement: Embedded Systems – Intern

Ananth Y R

**Objective:** The goal of this problem statement is to develop an embedded system that controls the switching speed of an LED, and the position of a servo motor based on the input frame pulse from a camera. The LED switching rate will be controlled in proportion to the input pulse width, while the servo motor’s position will be adjusted according to the number of LED "on" cycles.

**Solution:**

**Microcontroller Selection:**

The microcontroller selected is an ESP32. The microcontroller is selected due to the following reasons:

* High frequency clock (80 MHz) allows for processes to run within pulse being detected
* Has two cores allowing for parallel processing of interrupts and main function
* Has multiple inbuilt libraries and documentation to assist and allow for easier programming
* Has GPIO pins allowing for fast Input capture
* Easily accessible and prior experience with the hardware

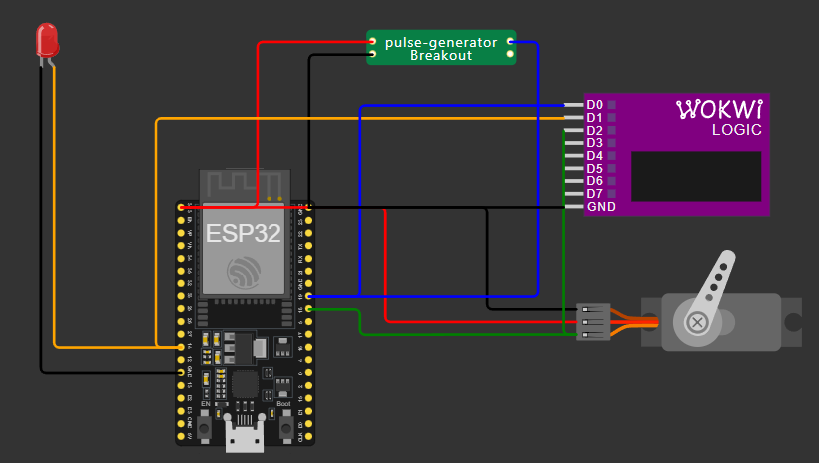
**Block Diagram:**

Figure 2. Diagram of circuit in simulation software

Figure 1. Block diagram of simulated circuit

(Simulated with a custom chip to generate pulses)

Camera

(Pulse input)

ESP32

(PWM Signal)

(Digital Signal)

LED

Servo motor

**Timing Diagram:**

Signals: logic.D0 – Pulse input from camera

logic.D1 – LED on/off signal

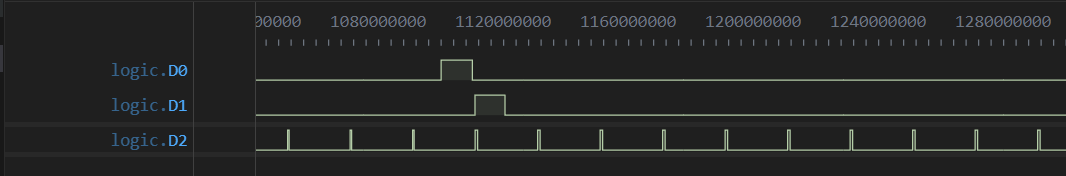
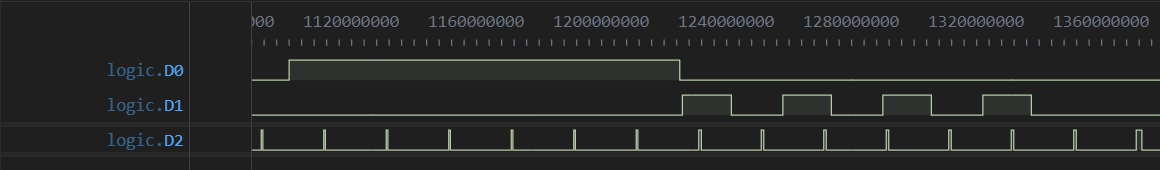
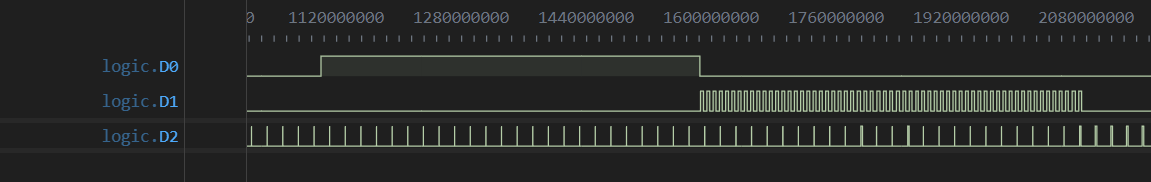
******** logic.D2 – Servo PWM signal

Figure 5. 10ms pulse

Figure 4. 125ms pulse

Figure 3. 485ms pulse

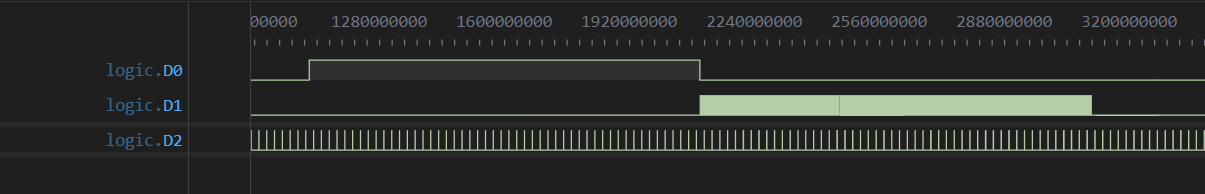
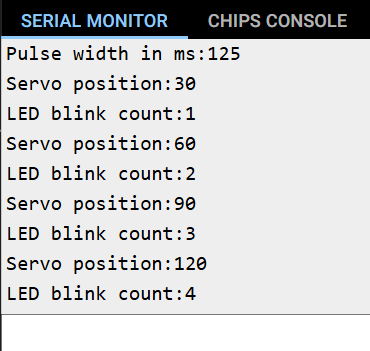
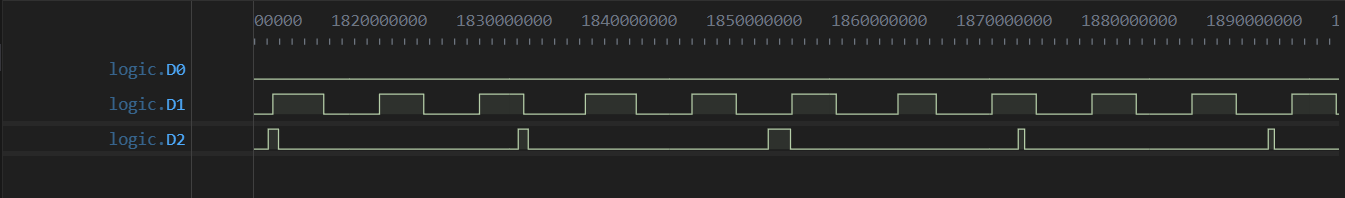
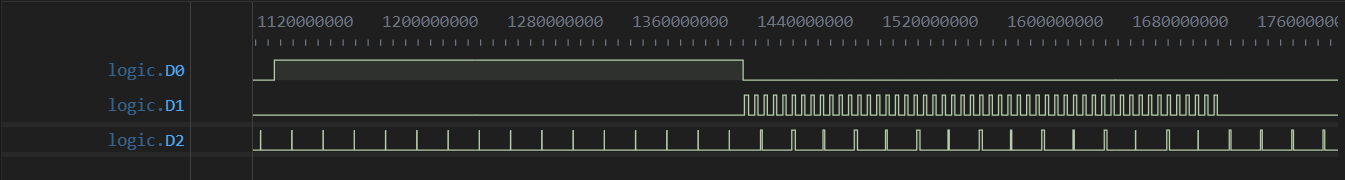
******Serial Monitor :**  **Considerations made:**

Figure 2. 1000ms pulse

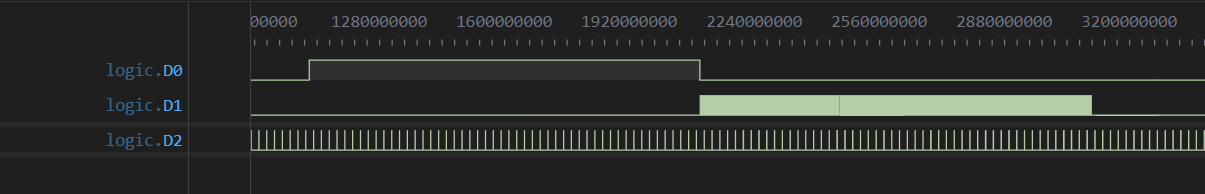
Figure 6. PWM signal (logic.D2) for the servo updates every 20ms (limited by the control signal frequency)

* Use of interrupts to allow new signal detection even when in LED blink sequence
* Updating Servo position every 20ms limited by 50Hz Servo control signal
* Limited use of delays, allowing for continuous new pulse detection
* Restricting input pulse values between 10-1000ms as specified

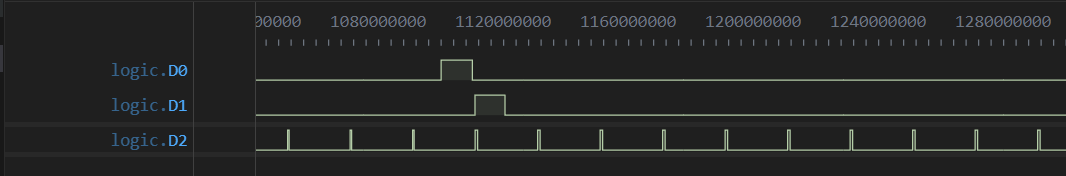
**485ms**

****

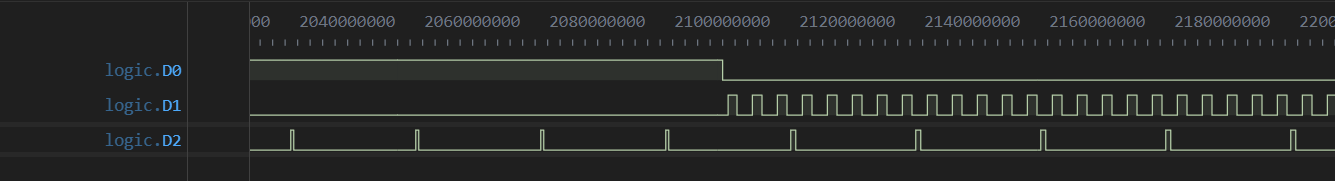
**300ms**

****

**1000ms**

****

**1ms**

****

**1000ms**