IV. Proposed new ES application.

In our project, An IR sensor in front of the gate checks that there is a car to open the gate, and there is another IR sensor behind the gate checks when the car moves from it, the gate closes and decreases the counter by 1 to show the number of free spaces in the garage on an LCD. When the car park in a free slot, there is another IR sensor that detect this car then will type that this slot is full on the LCD and will turn on a LED which is an indicator for filling this slot. When the car is going to exit the parking, there is an IR sensor in the front of the exit gate that will open the exit gate when a car was detected, then another IR sensor behind the exit gate will close the exit gate when a car was detected and will increase the counter of the free spaces on the LCD. In our parking there is a slot specified for disabled people, disabled people should activate a pushbutton before they park in these slots otherwise a buzzer will be activated, and error message will be typed on the LCD.

Hardware used components:

- 8 Infrared Sensors (2 at the entry gate, 2 at the exit gate and 4 at each slot)
- 2 servo motors for opening and closing the gates.
- An LCD that types the number of the free spaces and status of each slot
- A buzzer that will be activated if a car parked in a wrong slot without activating the pushbutton.
- 4 LEDs
- Pushbutton
- ATmega32

Software techniques used:

PWM for Controlling Servo Motors:

A microcontroller can easily control a servo; no extra driver, such as an h-bridge, is needed. To position the servo at any desired angle, only a control signal needs to be fed to it. The angle is controlled by the width of the positive pulse, which has a frequency of 50 Hz and a period of 20 ms.

We learned that SERVO MOTOR SG90 timings are as follows. The following numbers show the relationship between pulse width and servo angle. Keep in mind that these servos can only rotate between 0 and 180 degrees.

- 1.5 ms pulse = 0 degree.
- 2 ms pulse = 90 degrees. (Neutral position)
- 1 ms pulse = -90 degrees.
-

Servo motors can be managed by the PWM capability of AVR microcontrollers. In this manner, the CPU is free to perform other duties while the PWM automatically generates signals to lock the servo. You must have a fundamental understanding of hardware timers and PWM modules in AVR in order to set up and use PWM.

Here, we'll use the 16-bit AVR Timer1 Module with two PWM channels (A and B). The maximum frequency that the majority of AVRs can operate at is 16 MHz, which is the frequency of the CPU. So it is utilised in the majority of development boards, including xBoards and the Low Cost AVR Development Board. The prescaler was set to 8. So the timer will get 16MHz/8 = (0.5 uS period). We setup Timer Mode as Mode 14 whose features are:

- FAST PWM Mode
- TOP Value = ICR1 (input capture register)

Consequently, the timer will run from 0 to ICR1 (TOP Value). Below is a formula for PWM frequency and a computation for TOP value.

$$f_{OCnxPWM} = \frac{f_{clk_{I/O}}}{N * (1 + TOP)}$$

$$50hz = \frac{16Mhz}{8 * (1 + TOP)}$$

We therefore set ICR1A=39999, which provides a PWM period of 20ms (50 Hz). By correctly configuring bits COM1A1, COM1A0 (For PWM Channel A) and COM1B1, COM1B0, the Compare Output Mode is set (For PWM Channel B)

COM1A0 = 0 and COM1A1 = 1. (for PWM Channel A)
COM1B0 = 0 and COM1B1 = 1. (for PWM Channel B)

The OC1A (or OC1B) pin on Compare Match and SET (to high) at BOTTOM are both cleared by the adjustments. In ATmega16/ATmega32 chips, the PWM out pins are located on the OC1A and OC1B pins. This option provides a PWM output that is NOT inverted.

Now, the OCR1A and OCR1B registers can be changed to alter the duty cycle. The PWM high period is controlled by these two registers. The values needed for the following servo angles may be calculated since the timer's duration is 0.5uS.

Servo Angle 0 degrees require pulse width of 1.5 ms so value of OCR1A = 1.5ms/0.5us = 3000

Servo Angle 90 degrees require pulse width of 2 ms so value of OCR1A = 2ms/0.5us = 4000
Then, in order to control it, we require the values listed below using SERVO MOTOR SG90.

OCR1A=3000 should be set at 0 degrees.

OCR1A=4000 is set at 90 degrees.

This method allows you to determine the value of OCR1A (or OCR1B for a second servo) for any needed angle.

Timer 0 – CTC mode:

Set Point (SP) and Process Value were the two timer values we brought along (PV). We used to compare the process value with the set point after each iteration. The process value is reset when the process value reaches the set point or exceeds it. We've employed TIMER0. TIMER0 can count up to 255 because it is an 8-bit timer. In this case, we want the timer (process value) to reset as soon as its value reaches the set point of 125 or above.

In essence, the CTC Mode implements the same thing, it does so in hardware. This means that we are no longer concerned with constantly comparing the process value to the set point! This will not only prevent needless cycle waste but also guarantee improved precision.

As a result, the AVR CPU, which houses the hardware, is where this comparison occurs! A flag is set in the status register and the timer is immediately reset as soon as the process value reaches the predetermined point! CTC, or Clear Timer on Compare, is how the word is pronounced. As a result, all that is left to do is take care of the flag, which can be done considerably more quickly.
The timer's uint 8-bit counter is located in the TCNT0 Register. The Timer/Counter Register provides direct access to the Timer/Counter unit's 8-bit counter for read and write operations. The compare match on the following timer clock is blocked (removed) by writing to the TCNT0 Register. A comparative match between the counter (TCNT0) and the OCR0 Register could be missed if the counter is modified while it is running.
We'll focus on the TCCR0 Register's highlighted parts for the time being. By choosing the appropriate prescaler using these three Clock Select Bits (0,1,2), we configure the timer. Those responsible for the Waveform Generation Mode are bits 3 and 6. The maximum (TOP) counter value source, the type of waveform generation to be employed, and the counter's counting sequence are all controlled by these bits. The Timer/Counter unit supports three different operating modes: normal, clear timer on compare match (CTC), and two different pulse width modulation (PWM) modes.
Output Compare Flag 0 is bit 1 in the TIFR register. When there is a compare match between the data in the OCR0 - Output Compare Register0 and the Timer/Counter0, the OCF0 bit is set (one). While running the associated interrupt handling vector, hardware clears OCF0. A logic one can also be written to the flag to clear OCF0.

Interrupt:
The three external hardware interrupts for the AVR ATmega16/ATmega32 are labelled INT0, INT1, and INT2 and are located on pins PD2, PD3, and PB2. The ATmega controller interrupts whatever it is doing when one of these interrupts is activated and immediately begins the interrupt service function. Both level-triggered and edge-triggered external interruptions are possible. We can set up a trigger for this. While INT2 can only be edge-triggered, INT0 and INT1 can be level-triggered and edge-triggered.

The GICR register allows us to enable or deactivate external interrupts by setting Bit 6 – INT0 to 1 to Enable External Interrupt pin 0

MCUCR register is used to specify a level trigger or edge trigger on external INT0 and INT1 pins. IS01 and IS00 (Interrupt Sense Control bits). These bits specify what level or edge causes the INT0 pin to be triggered.

I/O PORTS:

BUZZER	A3
LCD_D4	A4
LCD_D5	A5
LCD_D6	A6
LCD_D7	A7
LCD_RS	B1
LCD_RW	B2
LCD_EN	B3
SLOT SENSOR 1	B4
SLOT SENSOR 2	B5
SLOT SENSOR 3	B6
SLOT SENSOR 4	B7
ENTERANCE SENSOR 1	C0
ENTERANCE SENSOR 2	C1
SLOT 2 LED(0)	C2
EXIT SENSOR 1	C3
EXIT SENSOR 2	C4
SLOT 3 LED(1)	C7
SLOT 1 LED(external)	D0
BUTTON	D2
SLOT 4 LED(2)	D3
EXIT MOTOR	D4
ENTERANCE MOTOR	D5