IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.15122

A Mass Air Flow (MAF)-Based System for Monitoring, Controlling and Optimizing Car Engine Operation, Using FPGAs and VHDL

Dr Evangelos I. Dimitriadis[1], Leonidas Dimitriadis[2]

Department of Computer, Informatics and Telecommunications Engineering,

International Hellenic University, End of Magnisias Str, 62124 Serres Greece[1]

Undergraduate student, Department of Information and Electronic Engineering,

International Hellenic University, 57400, Sindos Thessaloniki, Greece[2]

Abstract: A mass air flow (MAF) system, based on MD0550 sensor, FPGAs and VHDL, is presented here. The system is capable of providing a series of controls and subsequently activate respective alarm systems, related to car engine operation optimization. It can simultaneously monitor four basic air flow-related parameters. The first is air flow values lowering below lower critical set value, leading to respective insufficient fuel supply. Both blue LED and half left of board LEDs light up and buzzer also sounds to indicate the above fact with simultaneous white LED lighting, which represents additional fuel supply system activation. Second operation parameter checks if air flow values are within set limits and if this holds true, green LED lights up. Third basic operation parameter is related to air flow values becoming higher than upper set values, leading to red LED lighting. Finally, fourth basic operation parameter is checking whether air flow values remain above upper set value for a specific time period, leading to subsequent activation of decrease fuel supply system, represented with lighting yellow LED, while half right of board LEDs also light up. MD0550 air flow sensor used here and its output analog voltage values act as input to FPGA's ADC unit and converted values are presented to seven-segment displays. The system uses DE10-Lite FPGA board and taking into account that specific time periods, as well as upper and lower air flow limits can be set to a variety of values, gives our system the ability of implementation in a wide range of applications such as car engine operation, human breath detection, room occupancy detection, HVAC (Heating, Ventilation and Air Conditioning) system monitoring and weather stations. Our system is cheap to manufacture and can be also combined with IoT systems, allowing its use to logistics applications.

Keywords: MD0550 air flow sensor, air flow monitoring, FPGA, VHDL, Buzzer, LEDs.

I. INTRODUCTION

It is well known that FPGAs have attracted attention of researchers in the recent years, for industrial as well as for a variety of other applications. [1-14] FPGAs have the main advantage of combining software and hardware, thus enabling hardware programming for a series of applications. The most used languages for FPGAs' programing are VHDL and Verilog and VHDL is the one used in our work.

Although a lot of work has been done concerning implementation of FPGAs in a variety of applications as mentioned above, there are no works dealing with mass air flow sensor-based applications using FPGA boards. This kind of research can provide a car engine operation and optimization control, useful for IoT and logistics applications.

We present here a mass air flow sensor-based system, which can monitor air flow values and subsequently provide a series of controls in order to optimize engine operation due to direct relationship between air flow and fuel supply in most engines.

Our system can prevent unexpected engine shutdown due to fuel supply interruption and also prevent engine strain and wear and excessive fuel consumption due to over-running for a specific time period. The system can also inform user for normal air flow values, leading to normal engine operation.

Another benefit of our system is that it can work with a variety of air flow sensors which provide an analog output and also its cost is remarkably low.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.15122

II. DESIGN OVERVIEW AND OPERATION OF THE SYSTEM

Figure1 presents device overview and operational units of our system, using FPGA DE10-Lite board, while Figure 2 presents circuit diagram of the system. Subsequent Figure 3 presents the implemented mass air flow control system of this work.

It is obvious from the above figures that our system, except from DE10-Lite FPGA board, contains also some basic circuit parts. First is mass air flow monitoring system, second is output LEDs system and third is buzzer alarm system. DE10-Lite FPGA board used here offers, its seven-segment displays for presenting input air flow sensor voltage values and board LEDs mentioned above. Input voltage values have been multiplied by 100, for simplicity of presentation where a value of 100 corresponds to 1V.

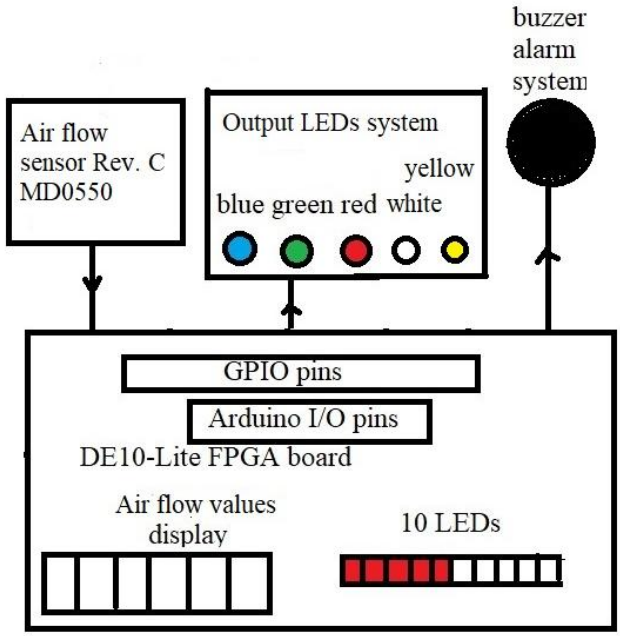

Figure 1: Device overview and operational units of our system.

Time is the other input value used here and it is provided by FPGA's clock.

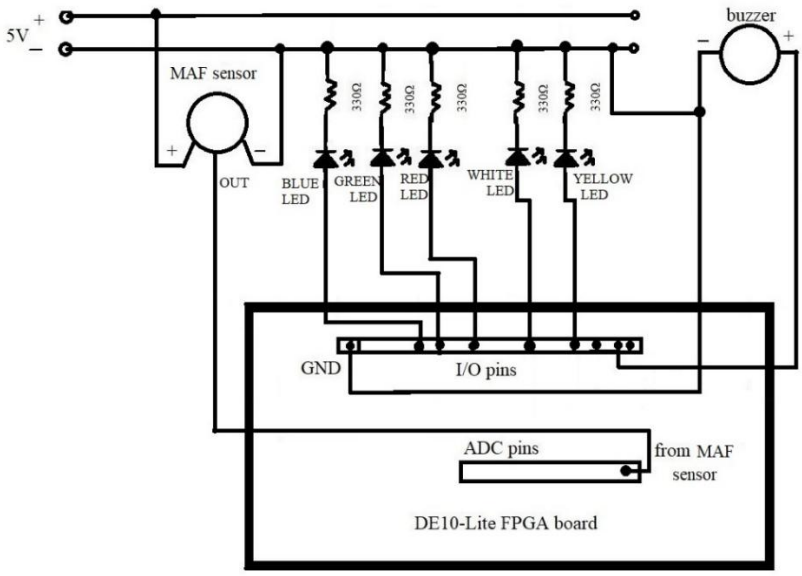

Figure 2: Circuit diagram of our system

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.15122

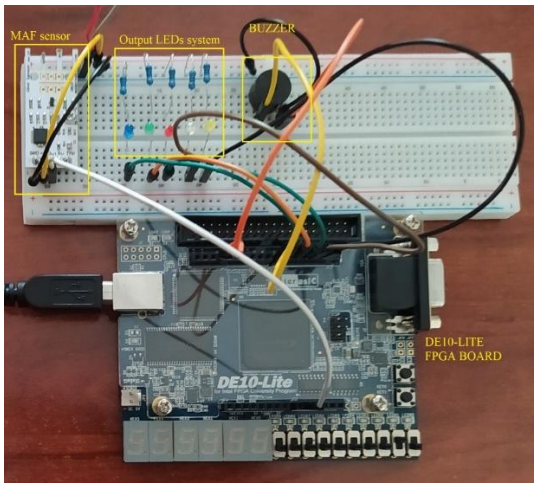

Figure 3: The implemented mass air flow control system of this work.

Our multiple mass air flow control system starts operating as soon as power supply +5V is applied to the circuits and the VHDL program is sent via USB Blaster interface, to FPGA chip. Input values from both MAF sensor unit and FPGA's clock are entered in our system. Analog to digital converter (ADC) of DE10-Lite proceeds to conversion and finally input voltage values are presented in seven-segment displays of the FPGA board. It must be mentioned here that mass air flow input voltage values (Vr) are divided into three basic zones shown below in Table1. The zones' limits are arbitrary chosen and they are related to MD0550 air flow sensor manual calibration.

Table 1: Characterization of air flow input values

Mass air flow input values Vr (V) (100 corresponds to 1V)	Characterization
Vr < 330	Low values
Vr >= 330 AND Vr < 400	Normal values
Vr >= 400	High values
Vr >= 400 for a set time period e.g. 5sec	Remaining high values

Upon our system's operation begins, simultaneously four additional to ADC conversion and displaying, basic processes begin to execute, according to Table 1. At first, air flow voltage input values are checked on whether they are in low values zone. If this holds true then simultaneously buzzer starts sounding, blue external LED lights up, left half of board LEDs light up and white LED is ON, representing additional fuel supply system activation, in order to maintain engine in operation, as shown in Figure 4.

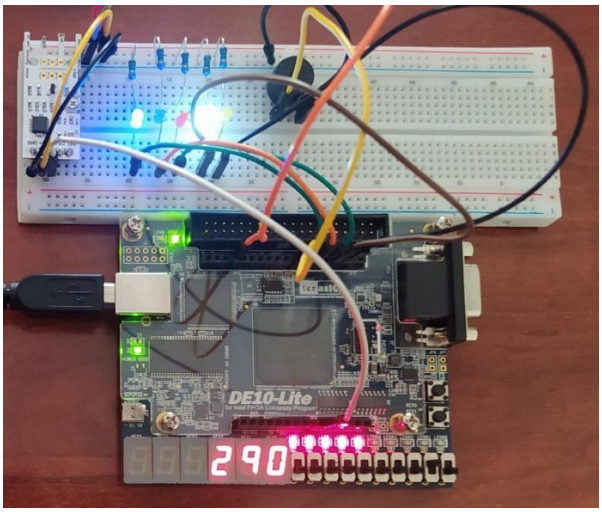

Figure 4: Air flow values below lower set values. Blue LED lights up and simultaneously white LED lighting, represents fuel supply system activation.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.15122

Second process is related to normal zone input air flow values and in this case green external LED lights up, presented in Figure 5.

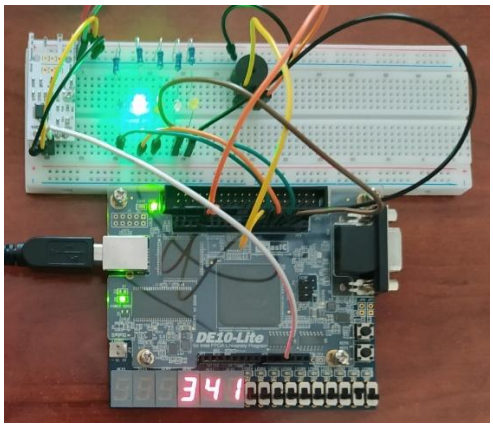

Figure 5: Air flow values within set limits and green LED is ON.

If input air flow values are checked, via third process, to be within high values zone, red external LED lights up indicating this fact, as shown in Figure 6.

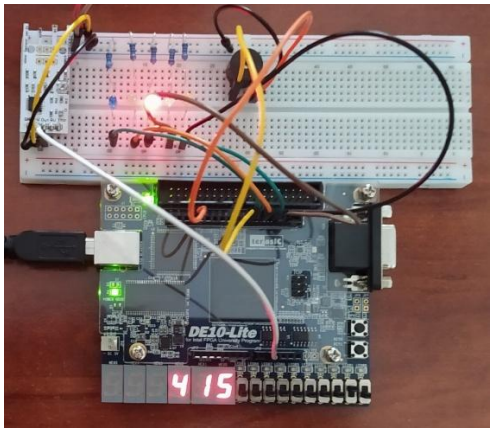

Figure 6: Air flow values above upper set limit values and red LED lights on.

Fourth basic process is checking whether mass air flow input values are continuously remaining within high zone. This situation poses a risk of damage to the engine and is also an unnecessary consumption of fuel, so yellow external LED and right half of board LEDs light up, with yellow LED representing the activation of fuel supply decrease system, as presented in Figure 7.

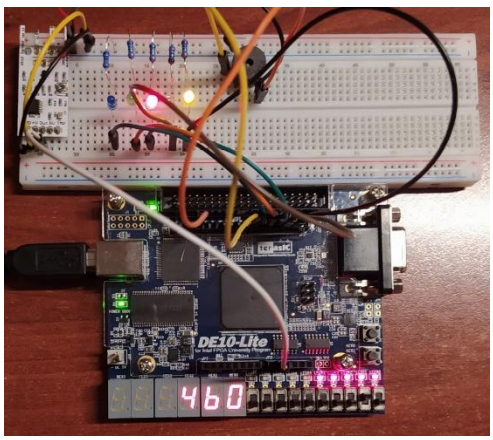

Figure 7: Air flow values remaining above upper set values for a set time period. Red and yellow LEDs are ON with yellow LED representing fuel supply decrease system activation.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.15122

Essentially we are using a counter unit which counts with clock pulses, the times that input air flow values are found to be above upper set limit. We must mention that, for simplicity reasons, we set the number of counted times exceeding upper set air flow input values, equal to 5 for a five seconds duration time period.
All of the above external LEDs are connected to I/O pins of the FPGA board and act as outputs for our system, capable of activating corresponding control systems.

It must be mentioned that specific time periods and mass air flow zones are set by the programmer. This gives our system the ability to monitor and control a variety of engines or other environments. In this work in order to obtain fast results we set time period of fourth zone to a range of 5-15 sec, as mentioned earlier.

Another important fact of our system's design is that it receives input air flow voltage values periodically, ensuring continuous air flow change monitoring.

III. PROGRAMING THE SYSTEM

We used Quartus Prime Lite Edition 21.1.1 to create the VHDL programs of our system. It must be mentioned here that before proceeding with the VHDL programming of our system, we had to set a series of parameters controlling the operation of DE10-Lite FPGA's Analog to Digital Converter (ADC). This converter plays a very important role in the whole system operation, since it converts the analogue input voltage from MD0550 air flow sensor connected to FPGA board to digital values, acting as main input of the system. The files created by the above ADC parameters setting are imported into the final project of our system.

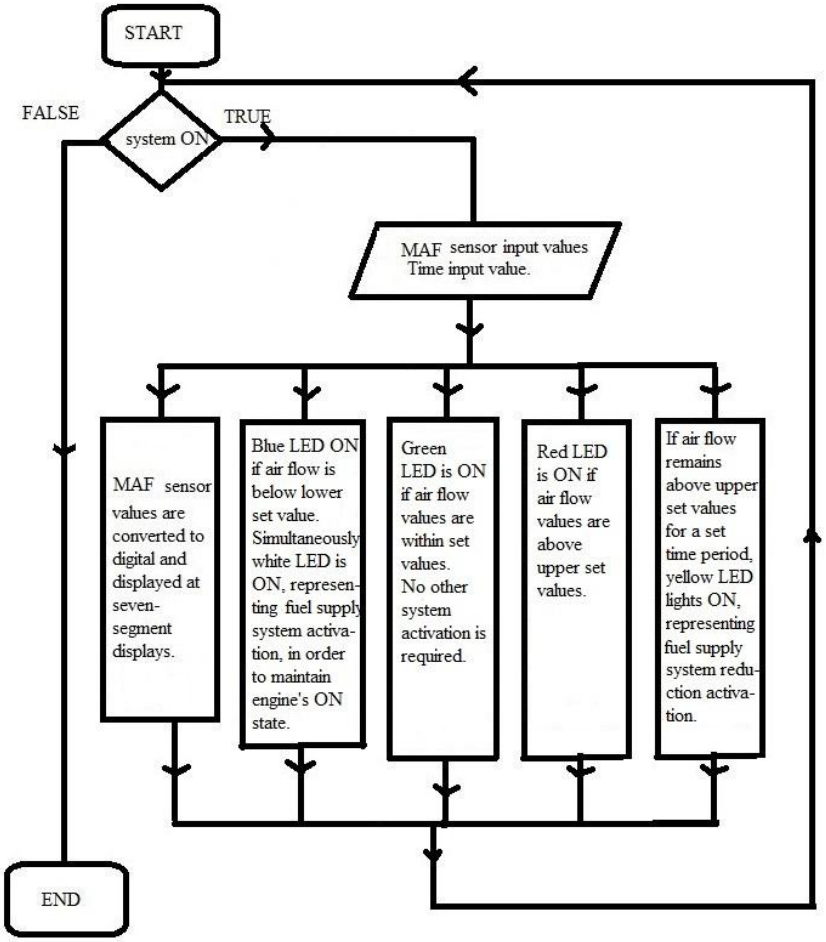

Figure 8: Flowchart diagram, presenting main functions-processes of our system.

A flowchart diagram, presenting main functions of our system is presented in Figure 8, while the APPENDIX contains the whole VHDL program.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026
DOI: 10.17148/IJARCCE.2026.15122

It is clear that the system basically operates five functions. All of them use processes in VHDL programming language. These processes are running as long as the system is ON, which is ensured when the system is powered and the switch SW1 of the FPGA board, which simulates the ignition switch of the car, gives binary 1 as logic output. All external or board's LEDs as well as buzzer system, are programmed to work as logic outputs, thus they are at the ON state if they receive binary 1 as logic output.

The first function playing definitive role in system operation, uses analog input voltage values provided from MD0550 mass air flow sensor as main input, convert them to digital values and present the final result in seven-segment displays. Calculated input sensor voltage values are used in many other processes, in order to activate control systems and external or internal LEDs. First function also includes time input values which come out of a separate process based on FPGA board's clock. It is a very important process whenever time parameter is needed.

Second function includes the process that simultaneously activates blue LED, buzzer system, left half of board LEDs and white LED in case that input air flow values are in the range of Vr< 330. Needless to say that our program gives the ability of setting air flow limits according to the application that our system is used.

Third function of our system refers to the procedure of checking whether input air flow values are within normal set limits, thus no other action is required than simply lighting green external LED. This means that engine works properly.

Fourth basic function of our system controls the unit responsible for monitoring air flow input values in order to determine if they exceed upper set limit. In this case red external LED lights up to inform user about this fact.

Fifth function of our system is programmed to count the number of times that air flow input values remain above upper set value for a specific time period also set by programmer. If the above number is equal or greater than specific value, for example 5 for 5 sec time period, then logic output 1 is send to red LED, right half of board LEDs and yellow LED. As mentioned above yellow LED represents fuel supply decrease system which is activated in order to reduce fuel consumption and also prevent engine's strain.

IV. CONCLUSION

A novel FPGA-based system is presented here, which manages to monitor mass air flow injected into an engine with subsequent injection of related fuel quantity and activate corresponding control systems for achieving optimum engine operation, which in our case are represented by internal or external LEDs and buzzer system. The system uses the MD0550 mass air flow sensor and it is capable of simultaneous monitoring whether air flow values are below lower set limit or within set range values or else above upper set limit. In the first case except from blue external LED, left half of board LEDs and buzzer, the system of additional fuel supply is activated, represented by white LED lighting. In the second case of air flow values being within set range, green LED lights up. Finally if air flow values are above upper set limit red LED is ON and in case that air flow values remain above upper set values for a specific time period, the system of decrease fuel supply is activated represented here by yellow LED, while right half of board LEDs also light up. Time period, lower and upper air flow values are simultaneously monitored and can be set to values that each application requires. This gives to our system the advantage of implementation to a variety of applications. The system can work with all commercial air flow sensors that provide an analog output and also it is easy to be manufactured, providing also the benefit of low cost.

ACKNOWLEDGEMENTS

We would like to express our warm thanks to **Theodora I. Dimitriadou, Eleni L. Fanara and Theodora E. Dimitriadou** for their support in completing the work.

REFERENCES

[1]. M. Yussup, M.M. Ibrahim, L. Lombigit, N.A.A. Rahman and M.R.M. Zin, "Implementation of data acquisition interface using on-board field-programmable gate array (FPGA) universal serial bus (USB) link", Advanc. in Nuclear Research and Energy Development, AIP Conf. Proc. 1584, 69-72 (2014), doi: 10.1063/1.4866106
[2]. A. Gujar, International Journal of Computer Science and Information Technologies, "Image Encryption using AES Algorithm based on FPGA", vol 5, (5), 2014, 6853-6859
[3]. S. Singh, A.K. Saini, R. Saini, I.J. Image, Graphics and Signal Processing, "Interfacing the Analog Camera with FPGA Board for Real-time Video Acquisition" 2014, 4, 32-38, DOI: 10.5815/ijigsp.2014.04.04

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026
DOI: 10.17148/IJARCCE.2026.15122

[4]. S. Muthukrishnan and R. Priyadharsini, International Journal of Computer Science and Mobile Computing, "32-Bit RISC and DSP System Design in an FPGA" vol 3, issue 12, Dec. 2014, pg. 361-368
[5]. L. Chen, Y. Chang, L. Yan, IEEE Transactions on Geoscience and Remote Sensing, "On-orbit real-time variational image destriping: FPGA architecture and implementation", 10.1109/tgrs.2022.3140428, 2022, pp. 1-1
[6]. S. Yarlagadda, S. Kaza, A. Tummala, E. Babu, R. Prabhakar, Information Technology in Industry, "The reduction of Crosstalk in VLSI due to parallel bus structure using Data Compression Bus Encoding technique implemented on Artix 7 FPGA Architecture", 10.17762/itii.v9i1.151, 2021, Vol 9 (1), pp. 456-460
[7]. C. Du, Y. Yamaguchi, Electronics, "High-Level Synthesis Design for Stencil Computations on FPGA with High Bandwidth Memory", 2020, 9(8), 1275; https://doi.org/10.3390/electronics9081275
[8]. X. Hao, C. Lin and Q. Wu, Electronics, "A Parallel Timing Synchronization Structure in Real-Time High Transmission Capacity Wireless Communication Systems", 2020, 9(4), 652; https://doi.org/10.3390/electronics9040652
[9]. P. A. Bawiskar, R.K. Agrawal, International Journal of Innovative Research in Science, Engineering and Technology, "FPGA Based Home Security System" vol.4, issue 12, Dec. 2015, p. 12865-12869, DOI: 10.15680/IJIRSET.2015.0412139
[10]. K. Saroch, A. Sharma, IOSR Journal of Electronics and Communication Engineering, "FPGA Based System Login Security Lock Design Using Finite
State Machine" vol 5, issue 3, Mar.-Apr. 2013, pp 70-75
[11]. R.S. Parikh, Int. Journal of Engineering Research and Application, "Alarm System Implementation on Field Programmable Gate Array" vol 8, issue 1, Jan. 2018, pp 01-04.
[12]. E. I. Dimitriadis and L. Dimitriadis, "A Simple, Low Cost and Multiple Input Alarm System, Functioning as Finite State Machine (FSM), Using VHDL and FPGAs", Journal of Active and Passive Electronic Devices, vol. 17, pp. 307–315, 2024.
[13]. E. I. Dimitriadis and L. Dimitriadis, "A g-sensor based alarm system, for multiple tilt sensor applications, using VHDL and FPGAs", Journal of Active and Passive Electronic Devices, vol. 18, pp. 119-129, 2024.
[14]. B. Prem Anand, C.G. Saravanan, "Development of research engine control unit using FPGA-based embedded control system", Journal of KONES Powertrain and Transport, Vol. 19, No. 3, 2012.

APPENDIX

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_signed.ALL;    -- step = step + 1
use ieee.std_logic_unsigned.ALL;
entity DE10_Lite_MAF_sensor is
generic(ClockFrequencyHz : integer:=50000000);
port
(
rst : in std_logic; --SW8
nRst    : in std_logic; -- Negative reset
Seconds : inout integer;
Ticks : inout integer;
led1: out std_logic;--1-5(0-4) LEDs light up when air flow values remain equal or higher than upper limit, for 5 sec
led2: out std_logic;
led3: out std_logic;
led4: out std_logic;
led5: out std_logic;
led6: out std_logic;--6-10(5-9) LEDs light up when air flow values are equal or lower than lower limit
led7: out std_logic;
led8: out std_logic;
led9: out std_logic;
led10: out std_logic;
led_blue: buffer std_logic; --air flow equal or below lower limit
led_red: buffer std_logic;--air flow equal or above upper limit
led_green: buffer std_logic;--air flow normal
led_blue_out: out std_logic;--air flow equal or below lower limit
led_red_out: out std_logic;--air flow equal or above upper limit
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.15122

```
led_green_out: out std_logic;--air flow normal
led_white: buffer std_logic;--air flow equal or below lower limit-automatic fuel supply increase system
led_yellow: buffer std_logic;--air flow equal or above upper limit for 5 sec-automatic fuel supply decrease system
led_white_out: out std_logic;--air flow equal or below lower limit-automatic fuel supply increase system
led_yellow_out: out std_logic;--air flow equal or above upper limit for 5 sec-automatic fuel supply decrease system

buzzer:out std_logic; --rings on air flow below  lower critical limits
Vr: buffer integer;
d2bbuf :buffer integer range 0 to 9;
d1bbuf :buffer integer range 0 to 9;
d0bbuf :buffer integer range 0 to 9;
SW0 : in std_logic;
SW1 : in std_logic;--car engine on/off switch

-- Clocks
ADC_CLK_10: in std_logic;
MAX10_CLK1_50: in std_logic;
MAX10_CLK2_50: in std_logic;
-- KEYs
KEY: in std_logic_vector(1 downto 0);
-- HEX
HEX0: out std_logic_vector(7 downto 0);
HEX1: out std_logic_vector(7 downto 0);
HEX2: out std_logic_vector(7 downto 0);

ARDUINO_IO: inout std_logic_vector(15 downto 0);
ARDUINO_RESET_N: inout std_logic);

-- GPIO
--GPIO: inout std_logic_vector(35 downto 0));
end entity;
architecture DE10_Lite_MAF_sensor_Arch of DE10_Lite_MAF_sensor is

-- Analog to Digital Converter IP core
component myADC is
port
(
clk_clk: in std_logic := 'X';
modular_adc_0_command_valid: in std_logic := 'X';
modular_adc_0_command_channel: in std_logic_vector(4 downto 0) := (others => 'X');
modular_adc_0_command_startofpacket: in std_logic := 'X';
modular_adc_0_command_endofpacket: in std_logic := 'X';
modular_adc_0_command_ready: out std_logic;
modular_adc_0_response_valid: out std_logic;
modular_adc_0_response_channel: out std_logic_vector(4 downto 0);
modular_adc_0_response_data: out std_logic_vector(11 downto 0);
modular_adc_0_response_startofpacket: out std_logic;
modular_adc_0_response_endofpacket: out std_logic;
reset_reset_n: in std_logic
);
end component myADC;
signal modular_adc_0_command_valid: std_logic;
signal modular_adc_0_command_channel: std_logic_vector(4 downto 0);
signal modular_adc_0_command_startofpacket: std_logic;
signal modular_adc_0_command_endofpacket: std_logic;
signal modular_adc_0_command_ready: std_logic;
signal modular_adc_0_response_valid: std_logic;
signal modular_adc_0_response_channel: std_logic_vector(4 downto 0);
```

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026
DOI: 10.17148/IJARCCE.2026.15122

```
signal modular_adc_0_response_data: std_logic_vector(11 downto 0);
signal modular_adc_0_response_startofpacket: std_logic;
signal modular_adc_0_response_endofpacket: std_logic;
signal clk_clk: std_logic;
signal reset_reset_n: std_logic;
type state_machines is (sm0,sm1, sm2, sm3, sm4);
signal sm: state_machines;
-- signals to store conversion results
signal ADCIN1,ADCIN4, ADCIN3,ADCIN2: std_logic_vector(11 downto 0);
signal AD1,AD4, AD3,AD2: std_logic_vector(11 downto 0);
-- signal for BCD digits
signal digit2b, digit1b, digit0b: std_logic_vector(3 downto 0);

signal digit2, digit1, digit0: std_logic_vector(3 downto 0);

-- signal to determine how fast the
-- 7-seg displays will be updated
signal cnt: integer;
signal state_LED_right: std_logic;
signal state_LED_left: std_logic;
signal state_Vr: integer;
signal xmax_maximum: integer;
begin
-- ADC port map
adc1: myADC port map
(
modular_adc_0_command_valid => modular_adc_0_command_valid,
modular_adc_0_command_channel => modular_adc_0_command_channel,
modular_adc_0_command_startofpacket => modular_adc_0_command_startofpacket,
modular_adc_0_command_endofpacket => modular_adc_0_command_endofpacket,
modular_adc_0_command_ready => modular_adc_0_command_ready,
modular_adc_0_response_valid => modular_adc_0_response_valid,
modular_adc_0_response_channel => modular_adc_0_response_channel,
modular_adc_0_response_data => modular_adc_0_response_data,
modular_adc_0_response_startofpacket => modular_adc_0_response_startofpacket,
modular_adc_0_response_endofpacket => modular_adc_0_response_endofpacket,
clk_clk => clk_clk,
reset_reset_n => reset_reset_n
);
clk_clk <= MAX10_CLK1_50;
reset_reset_n <= KEY(0);

-- process for reading new samples
p1: process(reset_reset_n, clk_clk)
begin
if reset_reset_n = '0' then
        sm <= sm0;
elsif rising_edge(clk_clk) then
        case sm is
                when sm0 =>
                        sm <= sm1;
                        modular_adc_0_command_valid <= '1';
                        modular_adc_0_command_channel <= "00001";
                when sm1 =>
                        if modular_adc_0_response_valid = '1' then
                                modular_adc_0_command_channel <= "00010";
                         ADCIN4 <= modular_adc_0_response_data;
                                sm <= sm2;
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.15122

```
                end if;
            when sm2 =>
                if modular_adc_0_response_valid = '1' then
                    modular_adc_0_command_channel <= "00001";
                    modular_adc_0_command_channel <= "00011";
                    ADCIN1 <= modular_adc_0_response_data;
                    sm <= sm1;
                    sm <= sm3;
                end if;
            when sm3 =>
                if modular_adc_0_response_valid = '1' then
                    modular_adc_0_command_channel <= "00100";
                    ADCIN2 <= modular_adc_0_response_data;
                    sm <= sm4;
                end if;
            when sm4 =>
                if modular_adc_0_response_valid = '1' then
                    modular_adc_0_command_channel <= "00001";
                    ADCIN3 <= modular_adc_0_response_data;
                    sm <= sm1;
                end if;
            when others =>
        end case;
end if;
end process;

-- process for conversion from binary to BCD (analog input voltage)
p3: process(AD2,d2bbuf,d1bbuf,d0bbuf)
variable vin: integer;
variable d2, d1, d0: integer;
begin
vin := to_integer(unsigned(std_logic_vector(to_unsigned(to_integer(unsigned(AD2)) * 500,
32))(31 downto 12)));
d2 := vin / 100;
d1 := vin mod 100 / 10;
d0 := ((vin mod 100) mod 10);
digit2b <= std_logic_vector(to_unsigned(d2, 4));
digit1b <= std_logic_vector(to_unsigned(d1, 4));
digit0b <= std_logic_vector(to_unsigned(d0, 4));
d2bbuf<= d2;
d1bbuf<= d1;
d0bbuf<= d0;
end process;

state_Vr<= (d2bbuf*100)+(d1bbuf*10)+(d0bbuf);
Vr<= state_Vr;

-- determine how fast the 7-seg displays will be updated
p4: process(reset_reset_n, clk_clk)
begin
if reset_reset_n = '0' then
cnt <= 0;
elsif rising_edge(clk_clk) then
if cnt < 20_000_000 then
cnt <= cnt + 1;
else
cnt <= 0;
AD1 <= ADCIN1;
```

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026
DOI: 10.17148/IJARCCE.2026.15122

```
AD2 <= ADCIN2;
AD3 <= ADCIN3;
AD4 <= ADCIN4;
end if;
end if;
end process;
process(MAX10_CLK1_50) is
   begin
      if rising_edge(MAX10_CLK1_50) then --ισοδύναμο του IF CLK'EVENT AND CLK='1'
         ------ If the negative reset signal is active
         if nRst = '0' then
            Ticks   <= 0;
            Seconds <= 0;
         else
            -- True once every second
            if Ticks = ClockFrequencyHz - 1 then
               Ticks <= 0;
                  Seconds <= Seconds + 1;
            else
               Ticks <= Ticks + 1;
            end if;
         end if;
      end if;
   end process;
process (Vr,seconds,MAX10_CLK1_50, state_LED_right, led_yellow,
xmax_maximum)
variable sec: integer;
begin
IF Vr=400 THEN--50000000 Ticks= 1sec and Vr=10 corresponds to 0.1V input
sec:=seconds;
end if;
IF (MAX10_CLK1_50'event and MAX10_CLK1_50 = '1') THEN
IF ((seconds>=sec AND seconds<=sec+5) AND Vr>=400) THEN
xmax_maximum<=xmax_maximum+1;
else
xmax_maximum<=xmax_maximum;
end if;
end if;
IF ((xmax_maximum>=5) AND Vr>=400) THEN
led_yellow<='1';
state_LED_right<='1';
else
led_yellow<='0';
state_LED_right<='0';
end if;
end process;
process (Vr, led_blue,led_red, led_green, state_LED_left, led_white)
begin
IF Vr>=400 THEN
led_red<='1';
else
led_red<='0';
end if;
IF Vr<330 THEN
led_blue<='1';
led_white<='1';
state_LED_left<='1';
else
```


IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.15122

```
led_blue<='0';
led_white<='0';
state_LED_left<='0';
end if;
IF Vr<400 AND Vr>=330 THEN
led_green<='1';
else
led_green<='0';
end if;
end process;
led_blue_out<=led_blue;
led_red_out<=led_red;
led_green_out<=led_green;
led_white_out<=led_white;
led_yellow_out<=led_yellow;
led1<=state_LED_right;
led2<=state_LED_right;
led3<=state_LED_right;
led4<=state_LED_right;
led5<=state_LED_right;
led6<=state_LED_left;
led7<=state_LED_left;
led8<=state_LED_left;
led9<=state_LED_left;
led10<=state_LED_left;
--buzzer sounds
Process(MAX10_CLK1_50,Vr)
variable i : integer := 0;
BEGIN
IF Vr<330 THEN -- buzzer sounds
if MAX10_CLK1_50'event and MAX10_CLK1_50 = '1' then
if i <= 50000000 then
i := i + 1;
buzzer <= '1';
elsif i > 50000000 and i < 100000000 then
i := i + 1;
buzzer <= '0';
elsif i = 100000000 then
i := 0;
end if;
end if;
end if;
end process;
process(SW0,digit2b,digit1b,digit0b)
begin
  IF SW0='0' THEN
  digit2 <= digit2b;--first digit of temperature
  digit1 <= digit1b;--second digit of temperature
  digit0 <= digit0b;--third digit of temperature
  end if;
end process;
WITH digit2 SELECT
HEX2 <= "01000000" WHEN "0000", -- display 0
"11111001" WHEN "0001", -- display 1
"10100100" WHEN "0010", -- display 2
"10110000" WHEN "0011", -- display 3
"10011001" WHEN "0100", -- display 4
"10010010" WHEN "0101", -- display 5
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.15122

```
"10000011" WHEN "0110", -- display 6
"11111000" WHEN "0111", -- display 7
"10000000" WHEN "1000", -- display 8
"10011000" WHEN "1001", -- display 9
"11111111" WHEN OTHERS; -- blank display

WITH digit1 SELECT
HEX1 <= "11000000" WHEN "0000", -- display 0
"11111001" WHEN "0001", -- display 1
"10100100" WHEN "0010", -- display 2
"10110000" WHEN "0011", -- display 3
"10011001" WHEN "0100", -- display 4
"10010010" WHEN "0101", -- display 5
"10000011" WHEN "0110", -- display 6
"11111000" WHEN "0111", -- display 7
"10000000" WHEN "1000", -- display 8
"10011000" WHEN "1001", -- display 9
"11111111" WHEN OTHERS; -- blank display

WITH digit0 SELECT
HEX0 <= "11000000" WHEN "0000", -- display 0
"11111001" WHEN "0001", -- display 1
"10100100" WHEN "0010", -- display 2
"10110000" WHEN "0011", -- display 3
"10011001" WHEN "0100", -- display 4
"10010010" WHEN "0101", -- display 5
"10000011" WHEN "0110", -- display 6
"11111000" WHEN "0111", -- display 7
"10000000" WHEN "1000", -- display 8
"10011000" WHEN "1001", -- display 9
"11111111" WHEN OTHERS; -- blank display
end architecture;
```