ANALYSING FUZZY LOGIC-BASED LINE FOLLOWING MODEL CAR

Boldizsár Könczöl[a], László Gál[b*]

[a] *ELTE, Faculty of Informatics, Computer Scientist MSc, I. year*
[b] *ELTE, Faculty of Informatics, Savaria Institute of Technology, associate professor*

ABSTRACT

In our previous work a fuzzy logic-based controller was successfully applied to a line following model car utilizing Arduino Uno. Regarding fuzzy operations (t-norms), this logic has several implementations and our aim was to show how functional can be the chosen ones, and whether there are any remarkable differences among them. The fuzzy rules were very easy to create, except the drastic t-norm, all of them completed the tests, thus it can be stated that using fuzzy logic is convenient for line following. In this paper we focus on the impact of using a more capable microcontroller (Espressif ESP32) based board for the controller. Imrpovement of results is expected because of the higher computing performance of this board.

Keywords: *fuzzy, Arduino, microcontroller, line following*

1. Introduction

Nowadays, self-driving cars became important, because they can navigate themselves without the driver's intervention. These cars have multiple sensors and cameras, which use different algorithms to analyze situations during driving and endeavor to make the right decision, while these take a huge amount of input signals and information into account. The classical Boolean algebra, the bivalent (binary) logic, is not effective for situations like this. There are several ways to implement not only bivalent logic but also polyvalent logic, for which fuzzy logic is an excellent example. It is made for giving back a value depending on multiple inputs or using these to send a signal to the controller output. It can be also calculated with using a gradually overlapping rulebase. Initially, the western countries thought that fuzzy logic is not suitable for practical applications, but on the contrary, eastern countries, especially the Japanese, were thinking the opposite way. They successfully simulated the use of fuzzy logic in railway transport in Senda, then they turned this simulation real in 1987 [1]. Today the system of fuzzy logic is capable of being used in different artificial intelligences, self-driving systems and of acting like human (or better than human) for several kinds of traffic situations. The everyday usage of fuzzy logic is not newfangled. This could be a surprise, but almost everyone used the concept of fuzzy, when for example say freezing, cold, warm, hot, burning for some objects. Using two different type of programmable microcontroller boards (Arduino Uno, ESP32 devkit v1) and some sensors connected to them a model car was built. This model car is capable of following the line drawn by the user. For decision-making procedure, the car uses fuzzy logic, fuzzy operations (t-norms), thus a fuzzy inference system was implemented. Furthermore, there is an interesting aspect in using fuzzy logic: it could be implemented by adapting different t-norms. Finally, during the comparison of test results, it was analysed that which t-norm

*Corresponding author: László Gál, laci.gal@gmail.com, gl@inf.elte.hu
https://doi.org/10.37775/EIS.2020.2.3

is more useful for line following or whether there are any improvements using ESP32 based board instead of Arduino Uno.

2. Methods

2.1. Fuzzy logic basics

To understand fuzzy sets, the knowledge of the classical set theory definitions and attributes could be useful. To distinguish these theories, the classical, non-fuzzy sets are called crisp sets. Definitions:

- Membership function: for any set X, a membership function on X is any function from X to the real unit interval [0,1].
- Fuzzy set: a fuzzy set is a pair $(U,\ m)$ where U is a set and $m : U \longmapsto$[0,1] a membership function [2].
- Membership value/degree: For an element x of X, the value $\mu_A(x)$ is called membership degree of x in the fuzzy set A.
- Support: $S(A) = Supp(A) = A^{>0} = x \in U|m(x) > 0$ is called support.
- Core: $C(A) = Core(A) = A^{=1} = x \in U|m(x) = 1$ is called core.
- Height: the supremum of a fuzzy set's values: $h(A) = sup_{x \in X} A(x)$.

2.2. Operations

Two of the three main operations (intersection, union, negation) are different from the classical (crisp) set operations. In fuzzy set theory there are multiple interpretations. The standards are Zadeh's operations [3].

- Intersection (t-norm): $(A \cap B)(x) = \min[A(x), B(x)]$.
- Union (t-norm): $(A \cup B)(x) = \max[A(x), B(x)]$.
- Standard negation: $\overline{A}(x) = 1 - A(x)$.

The intersection and union operations are associative and could be extended for unlimited number of fuzzy sets. This attribute is truly important, because in this project 3 inputs were analysed, thus we had to exploit this property on each t-norm. The union operation (t-conorm) was not used, but 6 well known or promising t-norms were tested in this application. The basic properties of these t-norms could be seen in following subsection.

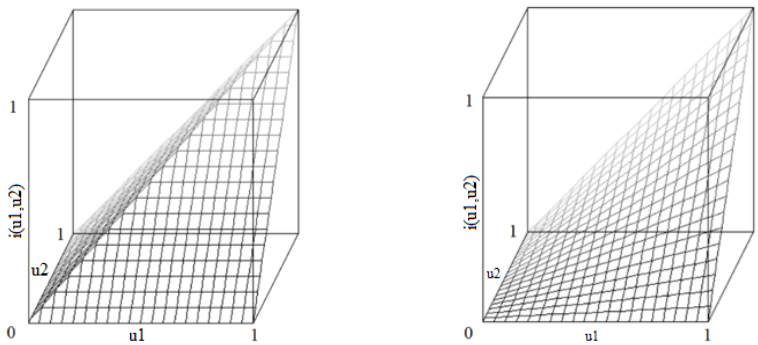


Figure 1. *a)* Graph of the minimum (standard) t-norm [4], *b)* Graph of the algebraic product t-norm [4]

2.3. Used t-norms

1. *Standard (minimum) t-norm*
   The minimum t-norm, as its name suggests, gives the minimum of the input values as output value (Fig. 1(a)). μ_1 and μ_2 are fuzzy membership values calculated from the two inputs (e.g. sensor values) using fuzzy membership functions:

$$w_m = \min(\mu_1,\ \mu_2). \tag{1}$$

   For three inputs:

$$w_m = \min(\mu_1,\ \mu_2,\ \mu_3). \tag{2}$$

2. *Algebraic (product) t-norm*
   The second test is on the algebraic product t-norm. The output is the product of input values (Fig. 1(b)):

$$w_a = \mu_1 \cdot \mu_2. \tag{3}$$

   For three inputs:

$$w_a = \mu_1 \cdot \mu_2 \cdot \mu_3. \tag{4}$$

3. *Drastic t-norm*
   An extremist t-norm, the output differs from zero only where at least one of the inputs is 1 (Fig. 2(a)). Due to its drastic behavior, difficulties were expected in applying this one for line following.

$$w_d = \begin{cases} \mu_1, & \text{if } \mu_2 = 1; \\ \mu_2, & \text{if } \mu_1 = 1; \\ 0, & \text{oterhwise.} \end{cases} \tag{5}$$

   For three inputs:

$$w_d = \begin{cases} \mu_1, & \text{if } \mu_2 = 1 \text{ és } \mu_3 = 1; \\ \mu_2, & \text{if } \mu_1 = 1 \text{ és } \mu_3 = 1; \\ \mu_3, & \text{if } \mu_1 = 1 \text{ és } \mu_2 = 1; \\ 0 & \text{oterhwise.} \end{cases} \tag{6}$$

4. *Łukasiewicz*
   Jan Łukasiewicz, polish mathematician's t-norm (Fig. 2(b)):

$$w_Ł = \max(0, \mu_1 + \mu_2 - 1). \tag{7}$$

   For three inputs:

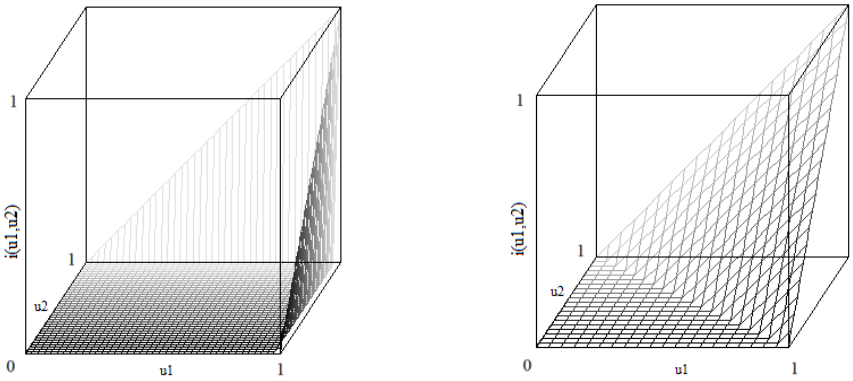
$$w_Ł = \max(0, \max(0, \mu_1 + \mu_2 - 1) + \mu_3 - 1). \tag{8}$$

5. *Trigonometric*
   A "smooth" t-norm based on trigonometric functions (Fig. 3(a)) [5]:

$$w_t = \frac{2}{\pi}\arcsin\left(\sin\left(\mu_1\frac{\pi}{2}\right)\cdot\sin\left(\mu_2\frac{\pi}{2}\right)\right). \tag{9}$$

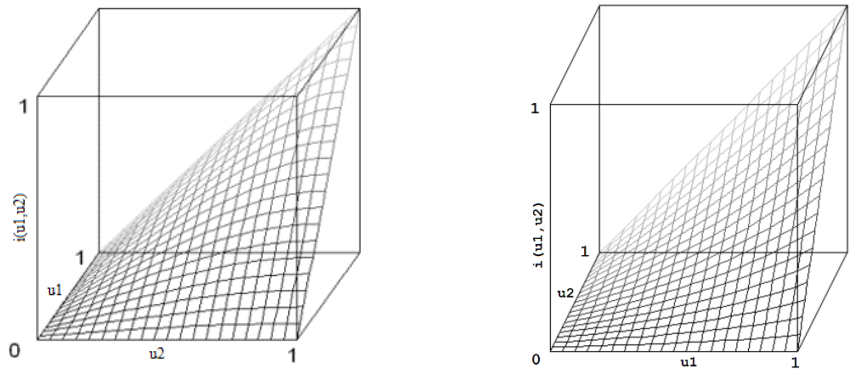
   For three inputs:

$$w_t = \frac{2}{\pi}\arcsin\left(\sin\left(\mu_1\frac{\pi}{2}\right)\cdot\sin\left(\mu_2\frac{\pi}{2}\right)\cdot\sin\left(\mu_3\frac{\pi}{2}\right)\right). \tag{10}$$



(a) (b)

Figure 2. *a)* Graph of the drastic t-norm, *b)* Graph of the Łukasiewicz t-norm [4]



(a) (b)

Figure 3. *a)* Graph of the trigonometric t-norm [5], *b)* Graph of the Hamacher product t-norm [4]

6. *Hamacher*
   The Hamacher t-norm has a parameter (ν), a specific variant of it ($\nu = 0$) is the Hamacher product t-norm (Fig. 3(b)):

$$w_H = \frac{\mu_1 \cdot \mu_2}{\mu_1 + \mu_2 - \mu_1 \cdot \mu_2}. \tag{11}$$

   For three inputs:

$$w_H = \frac{\mu_1 \cdot \mu_2 \cdot \mu_3}{\mu_1 \cdot \mu_2 + \mu_1 \cdot \mu_3 + \mu_2 \cdot \mu_3 - 2 \cdot \mu_1 \cdot \mu_2 \cdot \mu_3}. \tag{12}$$

3. Hardware setup

3.1. Main parts

1. Computing boards:

   Two different mainboards were used for the tests. The first was the quasi standard low-cost development board – Arduino Uno. The counterpart was an ESP32 based low-cost, Arduino IDE compatible development board with advanced performance and special features (like Wi-Fi and Bluetooth, although none of them was used for this project).

Arduino Uno (compatible) mainboard

- Atmel ATmega328P microcontroller,
- 8-bit architecture,
- 1 core,
- 16 MHz clock rate,
- 10-bit ADC,
- Arduino Sensor Shield v5.0.

ESP32 Devkit V1 mainboard

- Espressif ESP-WROOM-32 module,
- ESP32-D0WDQ6 microcontroller chip,
- 32-bit architecture,
- 2 cores,
- 240 MHz clock rate,
- 12-bit ADC.

2. L298N dual H-bridge DC motor controller.

3. 4 DC motors.

4. HC-05 (ZS-040) bluetooth module.

5. 3 MH-Sensor-Series infrared reflective sensors (with LM393 comparator).

6. The power is supplied by 3 pieces of 18650 battery cell, regulated by a DC-DC „step-down" converter module (9V stable).

ESP32 (microcontroller) based development boards are compatible with Arduino development environment (Arduino IDE), the programming language and the libraries are significantly the same for both boards. But as the specifications show us, the ESP32 is more powerful, than Arduino Uno (ATmega328P) and it has a floating-point unit while Uno not. According to these better results is expected from the ESP32, then from the Uno.

3.2. Modifications

1. *Infrared reflective modules*

   After connecting the three modules, the digitized values of the analog outputs of the modules were checked with the Arduino IDE serial monitor before the real tests. These values are between 0-1023 for Arduino (10 bit ADC) and between 0-4095 for ESP32 (12 bit ADC). There were huge differences between the infra sensors' values. The main reason for that was the difference in the (infra) LED light intensity of the modules. It could be easily checked this through a smartphone camera. Checking the sensors and their datasheets, to following conclusion was realised: the problem could be solved by changing the SMD resistors to variable ones (multi turn trimpot, trimmer). For safety, a 150 Ω resistor was connected in series for each trimmer. Because of the different required voltage level of the different boards (5V for Arduino Uno, 3.3V for ESP32), it was necessary to recalibrate the sensors for each mainboard.

2. *Power supply*

   The original power supply consisted of 4 pieces of AA batteries, but with these the speed control is very limited because of the 6/5.6 V voltage. These values are nominal battery output voltage, until this voltage reaches the motors, it goes through different electronic parts, so ca. 5.4/5 V usable voltage reaches the motors. Therefore, instead of this, 3 pieces of 18650 Lithium-ion batteries were used. Due to these batteries, the voltage is between 10.1 V and 12.6 V, which is more than enough. Contrary to this, the charging level changes results, so a DC-DC step-down module is connected to this pack of batteries. The output voltage was set to 9 V, so that the battery lasts longer, and the results will be consistent and easily comparable.

4. Results

The track was on a whiteboard and there were no crossroads. The car must have taken 3 full laps on the track. The time was measured with a stopwatch and from flying start, with an estimated precision of 0.1 s. In every case 3 measurements were carried out, and the median of these were compared. The results can be seen Table 1 and Table 2.

A Mamdani type fuzzy interference system was implemented with COG (center of gravity) defuzzification. While creating the rulebase, optimization was not an intended goal. The main aim was to test whether the operation could be affected, and to what extent do the usage of different t-norms affect this operation, especially if the fuzzy rulebase was created fast by an "expert" with minimal prior knowledge. The rules were created as described hereinafter.

If the left sensor is LIGHT, the middle sensor is LIGHT and the right sensor is LIGHT then LEFT motor STOP. In short form for the left motor: If {L, L, L}, then STOP. For better transparency, the 7 rules (R1-R7) for the left motor is given in the following formulae (LIGHT=L, DARK=D):

R1: If {L, L, L}, then STOP.
R2: If {L, L, D}, then FORWARD_MAX.
R3: If {L, D, D}, then FORWARD.
R4: If {L, D, L}, then FORWARD.
R5: If {D, D, L}, then BACKWARD.
R6: If {D, L, L}, then BACKWARD_MAX.
R7: If {D, D, D}, then BACKWARD_SLOWLY.

The rules for the right motor can be easily made based on these, just some swapping is needed with left and right side.

4.1. Arduino Uno test results

The L (light) and D (dark) trapezoidal shaped fuzzy membership functions [2] are implemented using the following parameters (each sensor sends analog signal converted to discrete values from 0 to 1023, the higher value is darker, these 4 values are the 4 breakpoints of the trapezoid):

L = {-1, 0 ,100, 700}
D = {100, 700, 1024, 1025}

In the expression $\{a, b, c, d\}$ a and d define the support, b and c define the core for the trapezoidal shaped (height 1) fuzzy membership function. In case of L = {-1, 0 ,100, 700} it was considered that the sensor output between 0 and 100 as "definitely light"; below -1 and above 700 as "not light at all"; whereas between 100 and 700 is a linear transition from "definitely light" to "not light at all". The higher the sensor output value, the darker the surface below.

The trapezoids, used in the consequent of the fuzzy rules, are originally set this way (range between 0 and 255, the higher value causes faster speed):

STOP = {-10, 0, 0, 10},
FORWARD_MAX = {60, 75, 80, 85},
FORWARD = {45, 55, 65, 75},
BACKWARD = {-80, -75, -70, -60},
BACKWARD_MAX = {-90, -85, -80, -70}.

The created rulebase's conclusions (the duty cycle of the motors: 0-255) are calibrated for a specific voltage (12 V). The value 55 is acceptable for 12 V, but this value with 6 V is not enough to start the motor. Thus, a "speed factor" value was created which scales this value (and all the other breakpoints of the consequence trapezoid), while the rulebase could stay the same.

According to the first measurements, the base "speed factor" was 1.92. After some tests it was experienced that the stable 9 V is appropriate, which has already been set in the step-down module. With this value, all the t-norms, except drastic could complete the tests. In these tests a slightly different speed factor of 2.0 was used for both mainboards. Performance of the Arduino Uno based fuzzy reasoning line follower model car was measured for each of the selected t-norms. Table 1 contains the results: measured time of three consecutive laps, where the only difference among the systems was the t-norm applied in the fuzzy inference system. As it was expected, utilizing drastic t-norm was not successful, at least not with this unoptimized fuzzy rulebase. The model car was not able to properly follow the line and stopped on the {L, L, L} condition all the time.

It is worthwhile to mention that although it was expected that the trigonometric t-norm based car perform very well, it proved to be the slowest one in this test. It was about 25% slower than the remaining four. Cause of this behavior was the higher computational resource requirement, considering it utilized much more floating-point operations (and function calls). Meanwhile, the others utilized mainly integer operations, only the Hamacher product performed one floating-point division operation. Of course, the COG defuzzification method used a few FP operations for all the t-norms.

Despite, Arduino Uno stores and handles double precision FP numbers (type double, 64 bit) as single precision ones (type float, 32 bit), the difference in the required computational effort is remarkable (compared to the integer operations). This way the latency, caused by the fuzzy reasoning, may affect the overall performance negatively.

4.2. ESP32 test results

The same Arduino IDE, the default Arduino IDE settings and almost the same source code was used for the ESP32 development board. In order to adapt to the different characteristic of the ESP32's analog-to-digital converter, some trivial changes had to be made to the fuzzy rules applied

Table 1. Results using Arduino Uno

t-norm	**3 laps time [s]**
Minimum	34.23
Algebraic product	36.95
Drastic	-
Łukasiewicz	37.72
Trigonometric	43.99
Hamacher product	33.41

Table 2. Results using ESP32

t-norm	3 laps time [s]	Improvement [%]
Minimum	32.49	-5.1
Algebraic product	28.25	-23.5
Drastic	-	-
Łukasiewicz	27.35	-27.5
Trigonometric	27.63	-37.2
Hamacher product	30.22	-9.5

for Arduino Uno. Each sensor's output is converted to data from 0 to 4095, the higher value is darker, these 4 values are the 4 breakpoints of the trapezoid:

L = {-1, 0 ,396, 3135}

D = {396, 3135, 4096, 4097}

The consequence trapezoids are the same. Performance of the ESP32 board based fuzzy reasoning line follower model car was measured for each of the selected t-norms. The same path and the same conditions were used as before (with Arduino Uno). Table 2 contains the following results: measured time of three consecutive laps, where the only difference among the systems was the t-norm applied in the fuzzy inference system.

Utilizing drastic t-norm was not successful, again. However, all the other t-norms benefited from the more powerful hardware. Although, the Łukasiweicz t-norm based system gained a notable performance boost, but in case of the trigonometric t-norm based system the improvement was huge. Therefore, these became the top two finishers – head to head.

4.3. Inference times

The fuzzy inference time of the reasoning was measured in both systems (Arduino Uno and ESP32) for all the selected t-norms using the Arduino built-in function millis(). Measuring the time for only one reasoning would be cumbersome, it would be in the millisecond and sub-millisecond range. We decided to measure the time of 1024 reasoning for both the left and right motors. This way Table 3 and Table 4 show the measured time required for $2 \cdot 1024$ fuzzy inference calculations for Arduino Uno and ESP32, respectively.

Not only the results have changed and improved by a factor of 10, but the movement of the car was way smoother with ESP32 than with Arduino Uno. Reducing calculation times contributed a great deal to the smoothness of the movement. Especially the trigonometric t-norm's performance improved a lot.

Table 3. $2 \cdot 1024$ calculations (Arduino Uno)

t-norm	interference time [ms]
Minimum	3 704
Algebraic product	3 758
Drastic	3 756
Łukasiewicz	4 035
Trigonometric	11 665
Hamacher product	4 448

Table 4. $2 \cdot 1024$ calculations (ESP32)

t-norm	interference time [ms]
Minimum	367
Algebraic product	378
Drastic	321
Łukasiewicz	344
Trigonometric	1 188
Hamacher product	471

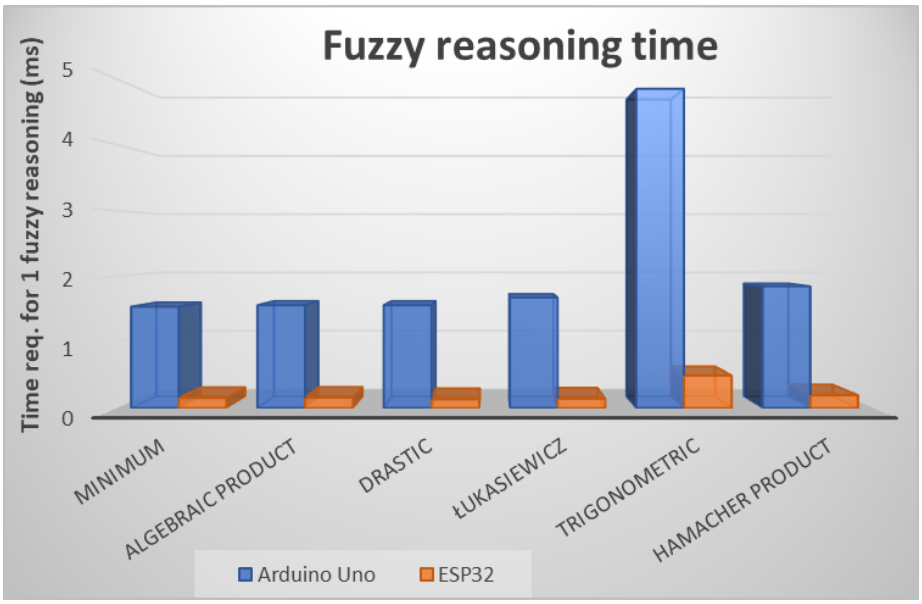


Figure 4. Time requirements for 1 fuzzy reasoning (Arduino Uno and ESP32)

4.4. Performance comparison

Fig. 4 depicts the performance improvements in the fuzzy reasoning time for different t-norms using ESP32 over Arduino Uno. Again, all the t-norm calculation times improved by approximately a factor of 10. Thus, the reasoning time moved from the millisecond to the sub-millisecond range, where the physical properties of the system (model car, mass, adhesion, DC motor inductivity) affect the performance more than the lag caused by the calculations. Therefore, the beneficial properties of a t-norm can prevail. Concerning the normalized reasoning time, the ratio among the t-norms are essentially the same (Fig. 5). While Atmel ATmega328P (Arduino UNO) does not have any floating-point unit so the FP calculations are made by software emulations, according to the specs, ESP32 does have FP units. It is a little bit disappointing to see that the FP operations performance improved only by the clock rate increasement. However, it is a bit unfair to compare these numbers because Arduino Uno handles "double" as "float", while ESP32 handles "double" as "double" despite ESP32

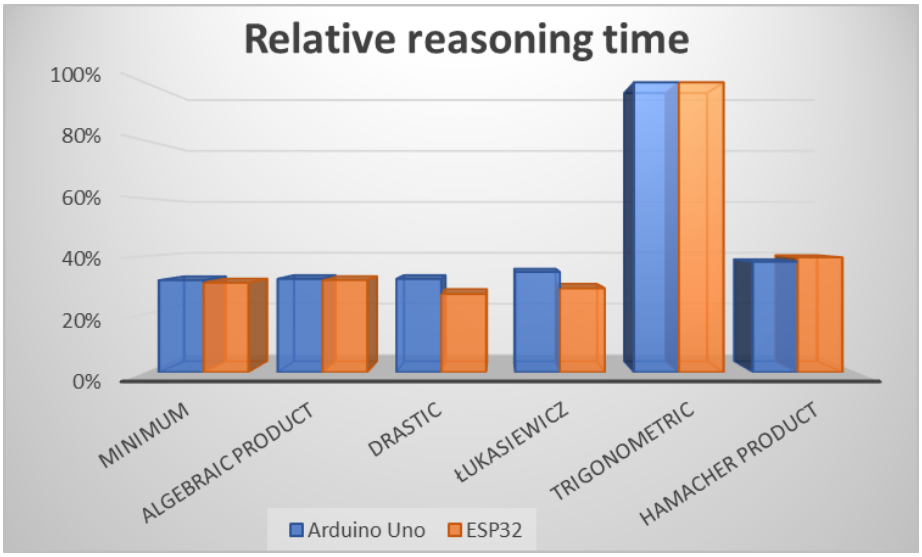


Figure 5. Normalized time requirements of fuzzy reasoning (Arduino Uno and ESP32)

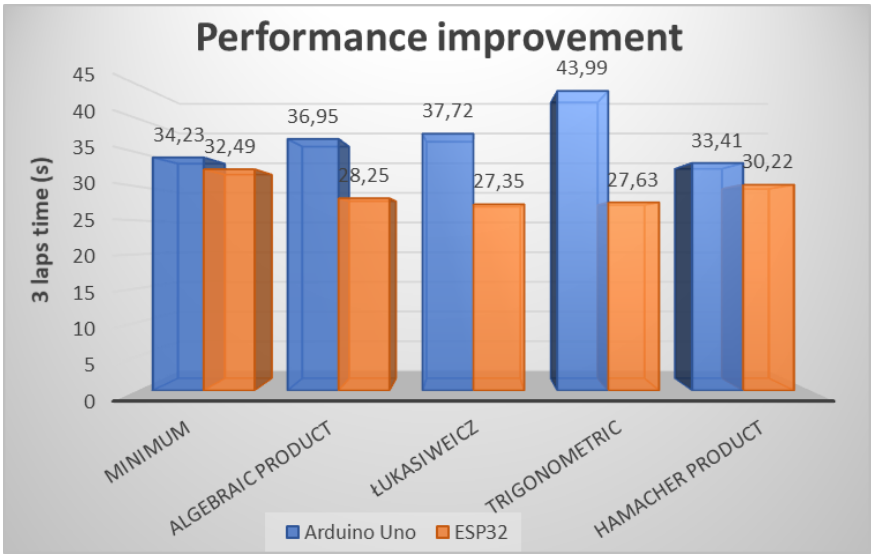


Figure 6. Overall performance improvements (Arduino Uno and ESP32)

has only single precision floating point units. More calculation time improvement for trigonometric t-norm was expected, its cause can be the strange behavior of the ESP32 FP units [6].

Fig. 6 depicts the overall performance improvements due to the application of the more powerful ESP32 instead of Arduino Uno. It is evident, the trigonometric t-norm based fuzzy reasoning line following model car benefited the most from the hardware change. Being the worst one executing on Arduino Uno (by a relatively large margin) it became the best, close together with the Łukasiewicz t-norm, 27.63 s and 27.35 s respectively.

5. Conclusions

Some speed increasement for the line following model car using the ESP32 board was expected in place of the Arduino Uno because of the higher computing capabilities. It became faster than it was before, also the line following was better. The average performance improvement was 20%, it varies between 5% (minimum t-norm) and 37% (trigonometric t-norm). We expected that the trigonometric t-norm based device performs very well, compared to the other t-norms based ones due to its smooth characteristic. Nevertheless, the Arduino Uno's MCU bottleneck will really hurt its performance, eliminating this obstruction enables to unfold its potential. Using the more capable board trigonometric and Łukasiewicz t-norms became the best of the tested t-norms in terms of "speed" for line following model cars.

Although, the trigonometric calculations still needed about 3 times more time than the other t-norms despite the fact that ESP32 has a dual core CPU that is supported by floating-point unit (FPU), it acts much better on ESP32 than on Arduino Uno. Just because a microcontroller includes an FPU, it does not mean the support of all types of floating-point operations. Therefore, we intend to extend the investigation involving commonly used microcontrollers like ARM Cortex-M4 MCU (based boards), more t-norms and optimizations.

6. Acknowledgement

The project has been supported by the European Union, co-financed by the European Social Fund EFOP-3.6.1-16-2016-00023.

7. References

[1] F. Oláh, *A Fuzzy Logika – Alapismeretek*, Autótechnika 05, 2009, pp. 42-43.

[2] Zs.Cs. Johanyák, Sz.Kovács, *A fuzzy tagsági függvény megválasztásáról*, A GAMF Közleményei, Kecskemét, XIX. évfolyam 2004, pp. 1-12.

[3] L.T. Kóczy, D. Tikk, *Fuzzy rendszerek*, Tyoptex, 2001, p. 14.,16.,18-21.,23.

[4] L. Gál, *Fuzzy modellek optimalizálása bakteriális típusú algoritmusokkal*, Ph.D Thesis, Győr, 2012

[5] L. Gál, R. Lovassy, L.T. Kóczy, *Function Approximation Performance of Fuzzy Neural Networks Based on Frequently Used Fuzzy Operations and a Pair of New Trigonometric Norms*, 2010 IEEE World Congress on Computational Intelligence, WCCI-2010, Barcelona, Spain, July 18-23 (2010), pp. 1514-1521.

[6] Espressif, *Unexpectedly low floating-point performance in C*, visited: 25.06.2020, url