Using Ultrasonic Breakpoint Sensors to Detect Transitions Between Rooms in Home Automation Systems

Christian Bodelsson

Contents

Abstract

This work is a subpart of a bigger project in home automation by monitoring each transition between rooms using ultrasonic sensors mounted to the doorways. The focus in this work is aimed at signalling a context server wirelessly by Bluetooth if a transition is made but there is also a stretch goal of distinguish the objects passing the doorway by looking at the time spent and the reflection area of the object. In this paper the system is presented by using a finite state machine model to illuminate the system characteristics. Additionally, an extensive evaluation of the system is given in this paper to evaluate the performance and reliability of the system.

Introduction

The main focus in this work will be to detect if an object has made a transition between rooms and at that time send a signal to a server indicating that a transition has been made i.e. keep track of the number of people in each room. This work is a subpart of a bigger project to automate a house with the help of gestures and movements. The main project is concentrating at helping disabled people with limited moving capabilities and constraints in their everyday life. It could be things that a person with the lack of disabilities do not think of such as pulling the curtains up or down, or toggling a light switch. Henceforth the project aims to explore the possibilities to help these peoples by making the house they live in, smart. The breakpoint implementation in this part of the work is needed in the smart house project since it is desirable to know how many people there are in each room. If the number of people is known by the system, the information can be used e.g. to help with the scheduling of the different peripherals attached to the system and save energy since all of the peripherals do not need to be active if there is no person in the room.

By using two ultrasonic sensors attached to each side of the doorway, the sensors will be used to keep track of people making transitions between rooms. The sensors that are used in this work are using ultrasonic pulses to measure distance [1] and these sensors are used in many areas like robotics and vehicle industry to calculate distance to objects within their environment [2] [3]. The technic used by the sensors is similar to bats and dolphin's echolocation, where the animal is emitting a high frequency sound to estimate the distance to an object [4] [5]. The sensors are emitting a high frequency tone called a ping. After the ping is generated by the sensor the sensor starts to record the time it takes for the ping to return back and since the speed of sound is known, it is a trivial task to calculate the reflecting objects position relative to the sensor. The speed of sound is determined by the medium the sound travels in and the temperature, humidity, pressure and frequency [6]. Since the breakpoint sensor consist of one pair of ultrasonic sensors (one on each side) the sensors can be used to identify which room the transition starts in and ultimately ends.

This work will also have a stretch goal of distinguish between objects passing the doorway since the surface and time may be distinguishable between objects e.g. A dog would in theory produce a different moving pattern between the doorway than a human in a wheelchair. This could potential help to filter out unwanted transitions between rooms since it is not desirable to detect e.g. a blind man's guide dog or other domestic animals.

Related Works

There have been a number of experiments using ultrasonic sensors. This section tries to outline what others have done in the area of home automation, object recognition and how they exploit the ultrasonic sensor capabilities in the mentioned area.

The main part of this work is to keep track of people making a transition between rooms using ultrasonic sensors as break point sensors. Other scientists have incorporated ultrasonic sensors into smart houses before with the job of indicating if people enters and leaves the room [7]. By using three ultrasonic sensors for three different rooms to identify the transitions between rooms and a 2.4GHz radio transceiver to communicate with a server. They achieved an average recognition rate of 81.3% by experimenting with three different individuals making an ideal entering and exiting routine between rooms [7].

One way to distinguish between objects and also to identify the human who is passing the ultrasonic sensor, is to look at the footstep signal on the ground produced by the human walking on different surfaces and by looking at different walking style. In [8] they can distinguish between different phases of the foot movements by looking at the Doppler effect [9] produced by the footsteps. By identifying heel strikes, toe slaps, and weight transfer they prove that they can identify the object passing the ultrasonic sensors [8]. In this work we only have two sensors to work with and therefore the work in this paper is not comparable with the work in [8].

It is also worth to mention that there is other works that combine the distance measuring feature of the ultrasonic sensors with neural networks to achieve a more accurate object recognition for more complex shapes [10]. In this work the authors are successfully able to distinguish between seven different PET bottles with high accuracy results. The setup conditions differ from the setup in this paper e.g. in [10] they use a sensor array containing a total of 28 ultrasonic sensors working as a multiparty unit to distinguish between shapes of different objects and therefore incomparable to this work.

When doing the classification of objects passing the breakpoint sensor it is important to look at the human traits given in this survey [11]. They look at different traits like weight, shape, motion, and vibrations caused by humans interfering with different kind of sensors, including the ultrasonic type.

There are also other ways to keep track of people inside a smart house as this paper tries to outline by pointing out the strengths and limitations with different types of sensors used in a smart house environment [12]

Methodology

The methodology is divided into different aspects depending on goal. First the design of the program is described to emphasise the software written for the breakpoint sensor. Secondly the stretch goal of distinguish between objects passing the doorway is described. Thirdly and lastly the communication and the breakpoint sensors location in the system is outlined to describe how the communication part will be handled.

Design Choices

Design considerations will be taken to ensure a stable and error correcting system i.e. the system is designed to recover from errors caused by erratic behaviour in the doorway e.g.

people who not complete their walk through the doorway. Furthermore, the system is able to self-reset if the system reaches an unrecoverable state e.g. halt of execution due to outer or inner errors.

To achieve a solid and modular system a state machine is developed to handle each of the systems states. The state machine is a natural design pattern in this case since the breakpoint sensor can easily be explained in terms of different states. This makes both the implementation easy to program but also to explain the system to the user.

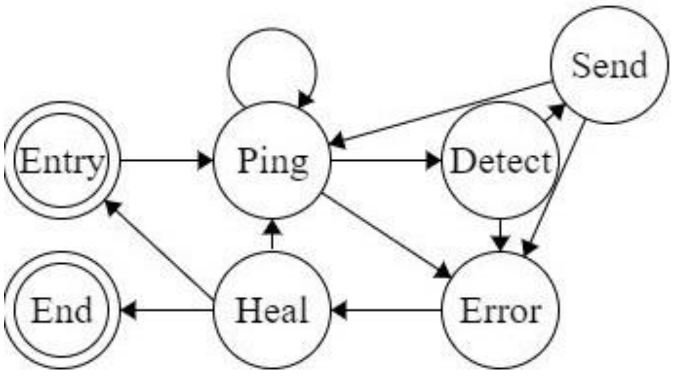

Figure 1 FSM design

As seen in Fig.1 the system is represented as a finite state machine (FSM) [13]. The first accept state is the entry state here the systems default parameters are initialized e.g. range of the sensors, tolerance, delays, timers, watchdog, all settings which is crucial for the system to function. This state also lets the program reset itself to its original state in case of an unrecoverable error. After the Entry state the program continue to the ping state and here the sensors will ping the environment until an error occurs or a detection event occurs. If an object is detected the ping state will return a detection signal and change state to the detection state. In the detection state the process of evaluating the object begins, in its most trivial implementation this detection state will forward the detection to the send state which sends the data to the Context server by wireless communication. But this state will also evaluate the passing object in terms of object classification. All of the above states could report errors i.e. instead of jumping to the intended state the program will jump to an error state which will try and handle the error by identify the error and send it to the heal state if the error is recoverable the heal state will resume the program, otherwise the program will reset the system by moving to the entry state. Errors could potentially also lead to a defined end state, but this is primarily for testing purposes during the implementation of the system. This design leads to a flexible system with the ease of adding further extensions/states to the system without adding complexity to the system.

Object Recognition

The system should strive to separate the humans from the other objects e.g. cats, dogs, flies, etc. i.e. tests with the goal of finding patterns in transitions between the sensors will be incorporated in this work. The range data from the sensor-pair will be extracted for different objects passing the door and analysed in Matlab to find distinguishable patterns between the objects passing the doorway. Specific patterns are expected between different kind of objects since they differ in area and time spent in the doorway note that this is exclusively focusing on actions that are considered normal behaviour when passing the doorway and not on erratic or undetermined behaviour in the doorway. Normal behaviour is the assumption that a person is passing the door in a walking action with a standard pace and the erratic behaviour is classified as jumping, somersaults, crawling, etc.

Communication

Since this implementation is not standalone but rather incorporated into a bigger system the system needs to be capable of communicating with a central server, called the Context. Since the sensors task is to decrement/increment people in the different rooms and not storing the actual people-count on the device, a protocol for sending the information to the Context must be established. This will be achieved by agree upon a suitable way of communicating with the Context group. Furthermore, it is also crucial to decide in conjunction with the Context group when to send the information since the Context may be busy, unresponsive, or in an unrecoverable state. The sensor is equipped with a Bluetooth radio module and thus able to communicate with the server.

Implementation and Results

In this section there will be an in-depth description of each implementation in the system. The first section will cover the hardware and software implementation and later moving to the deployment of the system in the real world.

Ultrasonic Sensor Implementation

In order to control the ultrasonic sensor a microcontroller from Atmel is used. The microcontroller is an 8-bit device with a CPU running at 16MHz and is called Atmega168pa. The microcontroller predominantly role is to calculate the distance using the ultrasonic sensors but also to forward the information wirelessly with Bluetooth technology.

The distance is calculated using a constant speed of sound 344m/s since the accuracy is not the main application in this work. The signal of the ultrasonic sensors is following an outlined schedule from its manufacturer.

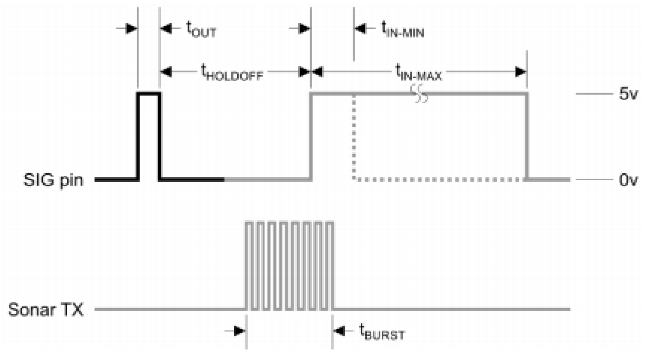

Figure 2 Ultrasound pulse [1]

The total time or the total width of the pulse echoing back is 18.5ms [1]. Therefor it is enough to use an 8-bit timer to measure the pulse width. But since the microcontroller is running at

16MHz the timer's frequency has to be scaled downed by a factor of 64 to produce a new frequency of the timer to be 250KHz (1).

$$C_f = \frac{F\_CPU}{P_{scale}} \ (1)$$

$$T_c = \frac{D}{C_p} - 1 \ (2)$$

$$C_p * (T_c + 1) = D \ (3)$$

This means that the timer will tick each ~4.02us and since the maximum value of an 8-bit timer is 255 this will subsequently produce a timer overflow each 1.024ms. This is can be considered to be enough of resolution since the minimum pulse width according to the data sheet is 115us [1]. When the timer overflows from 255 back to 0 a timer overflow interrupt will be triggered and an overflow counter will be incremented inside of the interrupt service routine. This means that the maximum amount of timer overflows that will occur in this implementation is limited to 18 since 18 * 1.024 = 18.43ms.

The datasheet specifies a maximum distance at 300cm and a minimum of 3cm [1]. In Sweden, the minimum width of a doorway should be more then 84cm so people with wheelchairs will be able to pass the door [14]. Therefore, the maximum length of the ultrasonic pulse will be 90cm but will be shorter in practice since an average human is about 50cm wide [15] so a length of ~50cm will be enough to detect if a human is passing the doorway, with the assumption that a person crossing the doorway will do so in a centred manner.

State Machine

The C programming language provide a lot of tools to design a robust state machine such as pointer to functions and the ease of designing advanced structures. In this work the state machine is implemented as suggested in [16] by using function pointers to resolve to the next state given by the previous state-return code which can be found by traverse a lookup table containing all possible transitions between the states.

Figure 3 Lookup table/Transition table

Source	**Return**	**Destination**
entry	ok	ping
entry	fail	end
entry	repeat	entry
ping	ok	send
ping	fail	end
ping	repeat	ping
send	ok	ping
send	fail	end
send	repeat	send

This is the foundation of the implementation with the state machine. Using the above transition table, the program is able of moving to different states depending on what the current state returned as its return code. E.g. if the entry state returns *ok* as a return code a transition from the entry state to ping state will occur but if the entry state returns fail then the program will halt however this will never occur since the final implementation will have error recovery states and added return codes to extend the above table.

Placement of Sensors

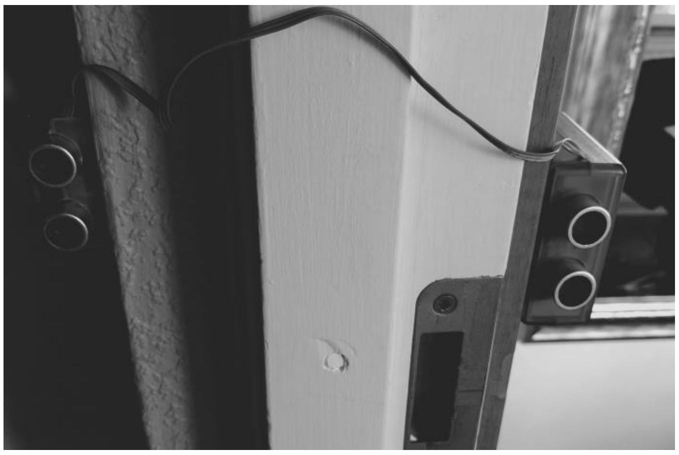

Figure 4 Sensor mounting in doorway

As seen in the above picture, the sensors will be attached in the doorway on each side of the wall giving the system abilities to detect which side detected the movement first and therefore provide an accurate registration of the object making the transition between two rooms. Also according to the ultrasonic sensors data sheet the sensors suffer from sound distortion if too close to the ground i.e. the sound can make extra bounces of the floor rending the measurement unreliable and result in a false detection of a transition. However, the minimum elevation above the floor is not mentioned in the data sheet but tests have concluded that there are no distortions above 1 meter and therefore the mounting position could be chosen arbitrary over 1 meter above the floor and under 1 meter below the roof.

Transitions

There are two kind of considerations that needs to be taken into account. The first is a simple transition between two rooms, room A to room B or vice versa.

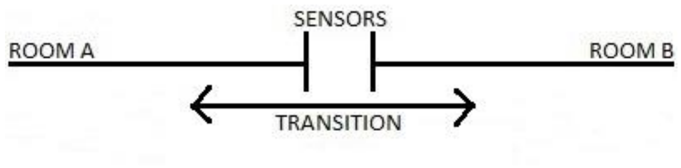

Figure 5 Transition between two rooms

In this illustration the person is moving from A to room B causing a signal to the Context server to decrement room A and increment room B. The above picture shows the first and main transition between rooms however there are another transition likely to occur which is the following.

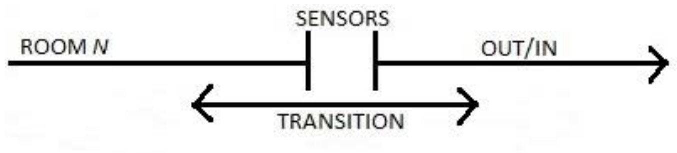

Figure 6 Transition from/to undefined state

In this illustration the person can move from room *n* in or out and also the other way around. This is a transition between a defined state and an undefined state e.g. A person who is coming from outside of the house into the room *n* is not making a transition as seen in Fig.5. Instead the transition should only produce an increment signal to the Context server and no decrement signal.

Context communication

The context server is implemented using Java and the GUI is built with a Java library called JavaFX [17]. The context server is responsible to update and present the number of persons in a room using the signals received from the break point sensors. The communication will be interrupt based since the break point sensors do not send data all the time but rather when the sensors have been crossed. The context server will therefore be alerted by the break point sensor that a sensor in room *n* has been crossed, thus giving the context the relevant information to keep the server updated. Furthermore, to manage the communication uniformly a protocol has been designed in conjunction with other peripheral groups in the smart house project. The communication is packet based and contains information about room, byte count, data, and a Cyclic redundancy check (CRC) error-detection code [18] to verify the integrity of the data transmitted.

Evaluation and Analysis

This part will present the performance analysis of the system in the context of the project specifications, meaning how well the implementation of the break-point sensor, work in the smart house.

Accuracy

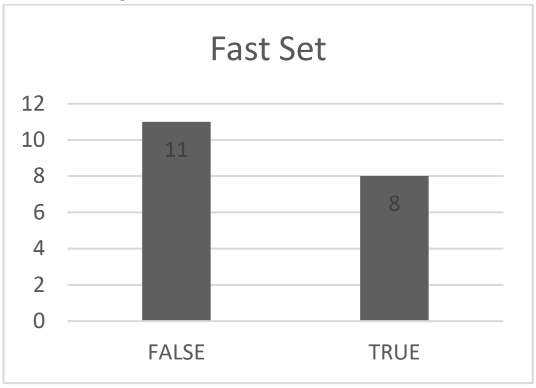

Figure 7 Fast speed long distance

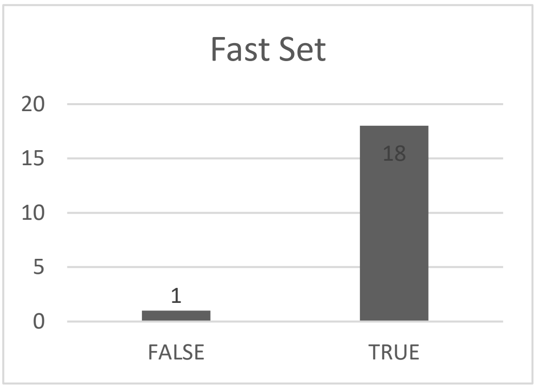

Figure 8 Fast speed short distance

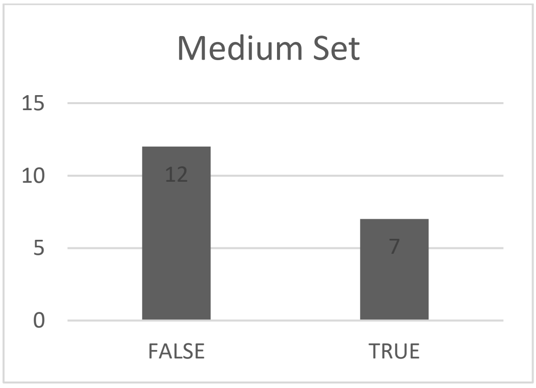

Figure 9 Medium speed short distance

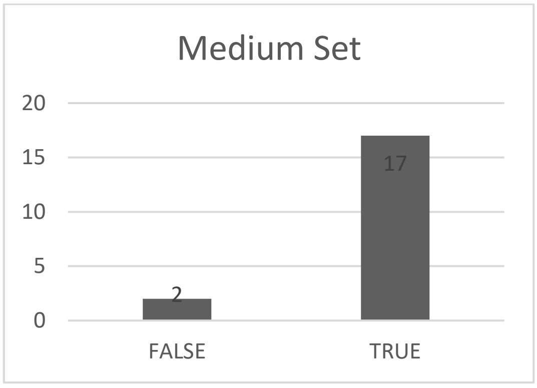

Figure 10 Medium speed long distance

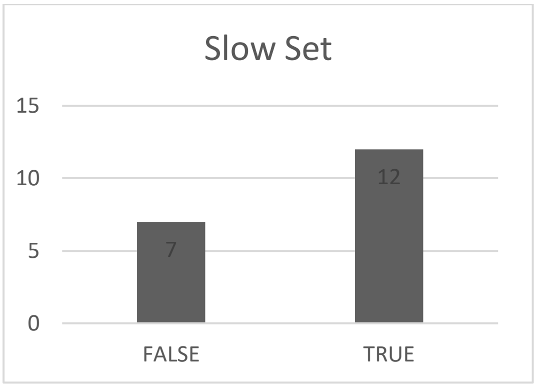

Figure 11 Slow speed short distance

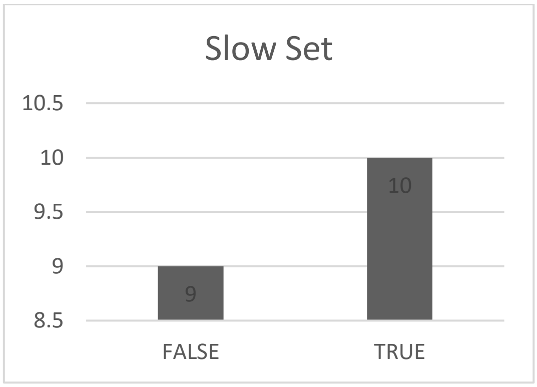

Figure 12 Slow speed long distance

Results of Effect

In order to evaluate the result of the break-point sensor, the N level, two factors, r repeat, N^2 *r model is used. In this model the delay between emitting an ultrasonic pulse is used as a factor. This factor has three levels, Low Speed 20ms, Medium Speed 110ms and High Speed 200ms. Another factor is the width of the doorway which acts as a workload for the system. The workload is divided into two levels, short distance and long distance. Each experiment is repeated 20 times, hence r = 20. The distance of the six states i.e. slow/short, medium/short is collected and then log transformed and presented then presented in Table 1 as an average of each set.

Table 1 Result of effects

	Workload	Slow	Medium	Fast	Row Sum	Row Mean	Row Effect
Distance	**Short**	1.31837	1.299047	1.34585	3.963267	1.321089	-0.2492455
	Long	1.803558	1.826762	1.828421	5.458741	1.81958	0.2492455
	Col Sum	3.121928	3.125809	3.174271	Sum of row sum: 9.422008		
	Col Mean	1.560964	1.5629045	1.5871355		Average of row mean: 1.5703345	
	Col effect	-0.0093705	-0.00743	0.016801			

Results of interaction

According to Table 1, Table 2 and according to the graphs in accuracy, slower speed increases the errors in the system. High speed with short distance workload achieves the highest accuracy compared to the other configurations.

Table 2 Interaction results

		Speed	
Workload	**Slow**	**Medium**	**High**
Short	0.0066515	-0.014612	0.00796
Long	-0.0066515	0.014612	-0.00796

Allocation of Variation

$$SSY = SSO + SSA + SSB + SSAB + SSE \quad (4)$$

A = number of column = 3

B = number of row = 2

R = replication = 20

U = row mean = 1.5703345

$SSY = \sum(log - transformed\ distance)^2 = 306.061$ (5)

$SS0 = ABRU^2 = 3 * 2 * 20 * 1.57033452^2 = 296.91405302683$ (6)

$SSA = BR\sum column\ effect^2 = 2 * 20 * [(-0.0093705)2 + (-0.00743)2 + (0.016801)^2] = 0.19701139085$ (7)

$SSB = AR\sum row\ effect^2 = 3 * 20 * [(-0.2492455)2 + (0.2492455)^2] = 8.69479831243$ (8)

$SSAB = R\sum Interactions\ result^2 = 20 * [(0.0066515)^2 + (-0.0066515)^2 + (-0.014612)^2 + (0.014612)^2 + (0.00796)^2 + (-0.00796)^2] = 0.19884458385$ (9)

$SST = SSY - SS0 = 306.061 - 296.91405302683 = 9.14694697317$ (10)

The SSE can be obtained either by computing individual errors or by using sums of squares as follows.

$SSE = SSY - SS0 - SSA - SSB - SSAB$ (11)

$SSE = 306.061 - 296.91405302683 - 0.19701139085 - 8.69479831243 - 0.19884458385 = 0.05629268604$ (12)

Explained by Speed: $100 * (SSA/SST) = 2.159\%$

Explained by distance: $100 * (SSB/SST) = 95.05\%$

Explained by Interaction: $100 * (SSAB/SST) = 2.17\%$

Unexplained: $100 * (SSE/SST) = 0.615\%$

Since the model explains a total of 99.379 % (2.159+95.05+2.17) variation, it seems to be a good model.

All three computed *F*-ratios (F-Computed) are higher than the values from the table. Thus, all three effects are statistically significant at a significance level of 0.10.

F-Table Computation

Significance level = 0.10

Speed

$$F\ [a - 1, ab\ (r - 1)]$$

$$DF\ (a - 1)\ =\ 2$$

$$DF\ ab(r - 1)\ =\ 114$$

Distance

$$F\ [(a - 1)\ (b - 1), ab\ (r - 1)]$$

$$DF\ (a - 1)\ (b - 1)\ = 2$$

$$DF\ ab(r - 1)\ =\ 114$$

Interaction

$$F\ [b - 1, ab\ (r - 1)]$$

$$DF\ (b - 1)\ =\ 1$$

$$DF\ ab(r - 1)\ =\ 114$$

Table 3 ANOVA Table

Component		**Sum of Squares**	**Percentage of Variation**	**Degrees of Freedom**	**Mean Square**	***F*–Computed**	***F*–Table [19]**
	SSY	306.061					
	SS0	296.91					
	SST	9.147	100.00	abr-1=120			
Speed	SSA	0.1970	2.15	a-1= 2	SSA/DF= MSA= 0.0985	MSA/MSE= 199.393	2.3497
Distance	SSB	8.6948	95.05	b-1= 1	MSB= 8.6948	17600.80	2.3497
Interaction	SSAB	0.1988	2.17	(a-1) (b-1) = 2	MSAB= 0.0994	201.2145	2.75
Error	SSE	0.0563	0.615	ab(r-1) = 114	MSE= 0.000494		

Confidence intervals for effects

$$Se = \sqrt{MSE} = \sqrt{0.0005} = 0.0224$$

SD of u:

$$0.0224 * \sqrt{1/abr} = 0.003$$

SD of Speed:

$$s_e \sqrt{a - 1/abr} = 0.0224 * \sqrt{3 - 1 \ /(3 * 2 * 20)} = 0.004$$

SD of Distance:

$$s_e \sqrt{b - 1/abr} = 0.0224 * \sqrt{2 - 1 \ /(3 * 2 * 20)} = 0.003$$

Confidence Interval

slow speed

$$call\,effect - 0.0093705 \pm 1.645 * 0.004 = -0.0093705 \pm 0.00658 = (-0.0027905, -0.0159505)$$

Medium speed

$$-0.00743 \pm 1.645 * 0.004 = -0.00743 \pm 0.00658 = (-0.00085, -0.01401)$$

Distance

$$-0.2492455 \pm 1.645 * 0.003 = -0.2492455 \pm 0.004935 = (-0.2443105, -0.2541805)$$

Since the confidence interval does not include zero, this effect is significant. The confidence intervals for various Speed and distance effects are listed in Table 4 below. The intervals are very narrow. This is because of replications caused by experimental errors.

Table 4 Confidence interval

Factor	Mean Effect	Standard Deviation	Confidence Interval
u (row mean)	1.5703345	0.003	
Speed			
Slow	-0.0093705	0.004	(-0.0027905, -0.0159505)
Medium	-0.00743	0.004	(-0.00085, -0.01401)
Fast	0.016801	0.004	(0.023381, 0.010221)
Distance			
Short	-0.2492455	0.003	(-0.2443105, -0.2541805)
Long	0.2492455	0.003	(0.2541805, 0.2443105)

Conclusion

By using the state machine, a stable implementation of the break-point sensor was achieved. The break-point sensors also integrated well into the smart house, by informing the main server when a person entered/exited the rooms. Also, it is not unexpected that the best configuration is the smallest delay and shortest distance. By keeping the delay small, the more responsiveness the system has and by keeping the distance short the room for missed/colliding pulses decreases. If the delay is big i.e. the time the next pulse emits is big enough, the sensors could fail to detect the person entering/leaving the room and thus lower the overall accuracy of the break-point sensor implementation. In this implementation and using this algorithm, it is crucial to keep the delay at a minimum level, as shown in the performance evaluation part.

Overall, the sensors work as intended in the smart house and it was a joy doing the implementation.

References

[1] Unknown, "Parallax," 9 11 2009. [Online]. Available: https://www.parallax.com/sites/default/files/downloads/28015-PING-Documentation-v1.6.pdf. [Accessed 10 02 2016].

[2] T. Q. H. S. J. C. K. L. JunHui Wu, "Environmental Modeling Based on Ultrasonic Ranging," in *Systems Engineering and Modeling* , Paris, 2013.

[3] J. H. H. L. B. L. Jun Liu, "An Ultrasonic Sensor System Based on a Two-Dimensional State Method for Highway Vehicle Violation Detection Applications," *sensors,* vol. 15, no. 10, pp. 9000-9021, 2015.

[4] Unknown, "scientificamerican," DIVISION OF NATURE AMERICA, 21 12 1998. [Online]. Available: http://www.scientificamerican.com/article/how-do-bats-echolocate-an/. [Accessed 10 02 2016].

[5] Unknown, "dolphins-world," BioExpedition, 10 01 2010. [Online]. Available: http://www.dolphins-world.com/dolphin-echolocation/. [Accessed 10 02 2016].

[6] A. J. Zuckerwar, "phy.mtu.edu," Academic Press, 01 01 2002. [Online]. Available: http://www.phy.mtu.edu/~suits/SpeedofSound.html. [Accessed 10 02 2016].

[7] S.-W. Lee, "A Room-Level Indoor Location System for Smart Houses," *Frontiers in neuroinformatics,* vol. 3, no. 11, 2009.

[8] A. E. E. James M. Sabatier, "Ultrasonic Methods for Human Motion Detection," in *Battlefield Acoustic Sensing for ISR,* Neuilly-sur-Seine, 2006.

[9] E. Adrian, "ncsa.illinois.edu," NCSA, 24 06 1995. [Online]. Available: http://archive.ncsa.illinois.edu/Cyberia/Bima/doppler.html. [Accessed 10 02 2016].

[10] M. B. Kozo Ohtani, "Shape Recognition and Position Measurement of an Object Using an Ultrasonic Sensor Array," in *Sensor Array,* Shanghai, InTech, 2012, pp. 53-67.

[11] G. D. A. S. THIAGO TEIXEIRA, "A Survey of Human-Sensing Methods for Detecting Presence, Count, Location, Track, and Identity," *ACM Computing Surveys,* vol. 5, no. 1, pp. 1-35, 2010.

[12] D. C. R. P. P. &. F.-P. L. Ding, "Sensor technology for smart homes," *Maturitas,* vol. 69, no. 2, pp. 131-136, 2011.

[13] Unknown, "ocw.mit.edu," MIT, 25 04 2011. [Online]. Available: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/unit-1-software-engineering/state-machines/MIT6_01SCS11_chap04.pdf. [Accessed 10 02 2016].

[14] Unknown, "Handikappförbunden," 01 01 2015. [Online]. Available: http://www.hso.se/PageFiles/99836/UT%20Faktablad%20Regler%20fo%CC%88r%20byggnader.pdf. [Accessed 07 03 2016].

[15] Unknown, “First in architecture,” First in architecture, 24 02 2015. [Online]. Available: http://www.firstinarchitecture.co.uk/average-male-and-female-dimensions/. [Accessed 07 03 2016].

[16] P. v. D. Linden, “Implementing a Finite State Machine in C,” in *Expert C programming Deep C Secrets,* Chicago, SunSoft Press, 1994, pp. 188-189.

[17] O. a. i. affiliates, “oracle,” oracle, 23 05 2014. [Online]. Available: http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784. [Accessed 07 03 2016].

[18] Unknown, “mathpages,” mathpages, 06 04 2010. [Online]. Available: http://www.mathpages.com/home/kmath458.htm. [Accessed 07 03 2016].

[19] D. Soper, “Critical F-value Calculator,” 01 01 2016. [Online]. Available: http://www.danielsoper.com/statcalc/calculator.aspx?id=4. [Accessed 20 04 2016].

Appendix

State Machine

```
/* [1] Assign the first temporary state as end
 * LOOKUP LOOP:
 * [1] If the current state equals a state in the table
 * and the return code match a return code in the table.
 * [2] Assign the new state to the temporary state
 * [3] Repeat if not found
 * [4] If found, break and return the next state
 */
static enum
state_codes lookup_transitions(enum state_codes current, enum ret_codes ret)
{
    int i = 0;
    enum state_codes temp = end;
    for (i = 0;; ++i) {
        if (state_transitions[i].src_state == current &&
            state_transitions[i].ret_code == ret) {
            temp = state_transitions[i].dst_state;
            break;
        }
    }
    return temp;
}


/* Main function/ main loop checks just for the "end state" else continuing to the next state given by the
transition function above.
*/


Int

main(void)

{
    enum state_codes cur_state = ENTRY_STATE;
    enum ret_codes rc;
    uint8_t (* state_fun)(void);

    for (;;) {
        state_fun = state[cur_state];
        rc = state_fun();

        if (EXIT_STATE == cur_state)
            break;
        cur_state = lookup_transitions(cur_state, rc);
    }
    return 0;
}
```

Calculate Distance

```
/* C_CPU / 64 = 250kH one clock cycle 1/250kH = 4us
 * Resolution 4us*(1+255) = 1.024ms/255 = 4.016us each tick
 * The width of the pulse per centimetre is according to the datasheet
 * 29.033us/cm so maximum width of 300cm would be 29.033 * 600cm = 17.41ms        round
 * trip time yielding an upper limit of elapsed time to 17 * 255 + (255 * 0.41) =
 * 4439
 * minimum width of the pulse would be 29.033 * 4cm = 1.161us yielding a lower
 * limit to (1.024ms/0.1161ms) = 255 / 8.81 = ~28
 * Speed of sound depends on temperature E.g. C = 331.5 + (0.6 x Tc) m/s
 * Measure by pulling, not exact for measurements but for our implementation
 * this is enough.
 *
 */
static void
echo(double *ping_value,const uint8_t pingpin)
{
   /*Round-trip-time, unit microseconds*/
   uint16_t elapsed_time;
   /* ------Trigger Pulse--------------------------*/
   PORT_ON(DDR, pingpin);
   PORT_OFF(PORT, pingpin);
   _delay_us(2);
   PORT_ON(PORT, pingpin);
   _delay_us(5);
   PORT_OFF(PORT, pingpin);
   /*--------End Trigger Pulse---------------------*/
   /*--------Meassure pulse------------------------*/
   FLIP_PORT(DDR, pingpin);
   loop_until_bit_is_set(PIN, pingpin);
   reset_timer_0();
   loop_until_bit_is_clear(PIN, pingpin);
   /*--------Stop Meassure pulse-------------------*/

   /* MAXCOUNTER is dependent on timer */
   elapsed_time = (tot_overflow * MAXCOUNTER) + TCNT0;

   /*Simple band-pass filter*/
   if (elapsed_time >= ping_distance_max)
      *ping_value = 0;
   else if (elapsed_time <= ping_distance_min)
      *ping_value = 0;
   else
      *ping_value = (elapsed_time * TO_CM);
   _delay_us(250);

}
```

Detect Transition

```
/* The Ping sensor sends a trigger pulse of 5us, which makes the ping sensor
 * send out an ultrasonic burst of 40kHz for 200us. Then we wait for the pulse
 * to come back.
 * Since the implementations is targeting slow humans, some toughness is
 * introduced in the system, by using _delay functions (just count clock,cy).
 */
uint8_t
ping_state()
{

   double sensor_A_val,sensor_B_val;
   uint8_t sa = 0;
   uint8_t sb = 0;

   echo(&sensor_A_val,PINGPIN_A);
   echo(&sensor_B_val,PINGPIN_B);


   if(sensor_A_val <= x_cm && sensor_B_val <= x_cm){
      return repeat;
   }else if(sensor_A_val <= x_cm){

      loop_until_sensor_is_unblocked(&sensor_A_val,PINGPIN_A);
      loop_until_sensor_is_blocked(&sensor_B_val,PINGPIN_B);
      loop_until_sensor_is_unblocked(&sensor_B_val,PINGPIN_B);

      evt = SENSOR_A_EVT;
      return ok;

   }else if(sensor_B_val <= x_cm){

      loop_until_sensor_is_unblocked(&sensor_B_val,PINGPIN_B);
      loop_until_sensor_is_blocked(&sensor_A_val,PINGPIN_A);
      loop_until_sensor_is_unblocked(&sensor_A_val,PINGPIN_A);

      evt = SENSOR_B_EVT;
      return ok;

   }else{
      _delay_ms(20);
      return repeat;
   }
   return fail;
}
```