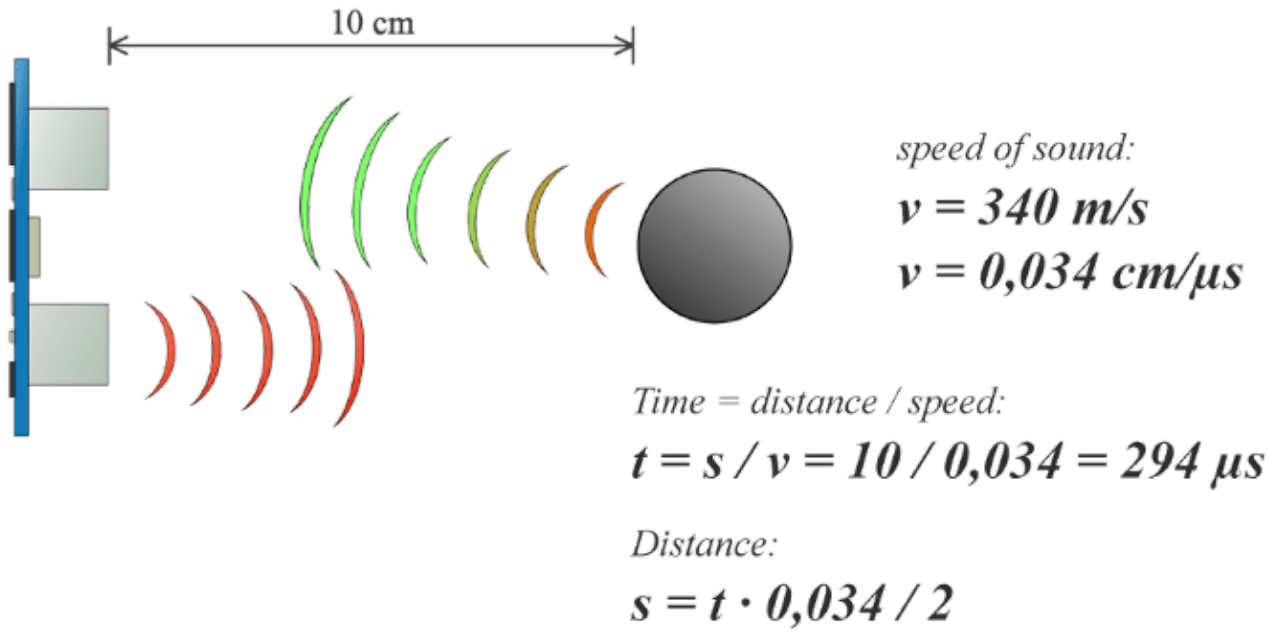
Israel N. Gbati

Follow 41 Followers About



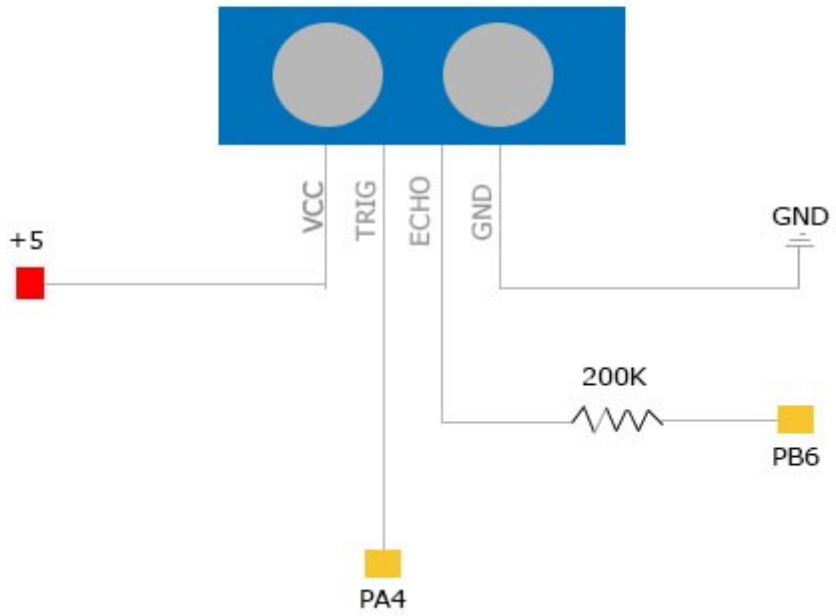
-source howtomechatonics.com

— On Writing Drivers for the HC-SR04 Ultrasonic Sensor

Israel N. Gbati Aug 14, 2017 · 5 min read

The HC-SR04 Ultrasonic sensor is a very popular low cost ultrasonic sensor used for non-contact distance measurements. The module comes with a separate transmitter and receiver. To determine the distance of an obstacle in front of the module, the transmitter (TRIGGER) automatically transmits **eight 40Hz** ultrasonic bursts/signals. **The bursts hits the obstacle and then bounces back to the module**. When/if the ultrasonic signal gets back, the receiver (ECHO) pin automatically turns HIGH for a duration of the sum of time taken to send and receive the ultrasonic burst. The time is then multiplied by a constant to get the distance.

2. Wait for signal to bounce back (ECHO)
3. Start Timer when ECHO is detected i.e. Echo is HIGH
4. Stop Timer when ECHO pin goes LOW
5. Read Timer Value.
6. Convert value to distance



pinout

Specification Rating

Working Voltage -DC 5V

Working Current -15mA

Maximum Range -4m

Working Frequency- 40Hz

Trigger Signal — at least 40us

How do you detect the ECHO ?

Remember from the pinout above, we have set the Echo pin as an input pin, so to detect the ECHO we can simply check whether the state of the input pin is HIGH or not.

In the circuit diagram above, we have connected the Echo pin to PB6 of our microcontroller. To detect if PB6 is HIGH and then start a Timer :

1. We can check with a simple logical comparison and then keep incrementing a counter variable while the condition is true

as simple as

```
//Pseudo code

while(PB6!= 0){

counter++

}
```

distance = counter * CONSTANT

And then converting the value of counter to distance by multiplying it by a constant. This method is straight-forward but not very precise.

2. **The more precise and efficient way, which I highly recommend is by setting a General Purpose Timer to detect the rising and falling edges of PB6. That way you can detect exactly when PB6 gets HIGH and when it goes LOW**.

Getting Your Timers Right

To write a more robust piece of code to interface with the ultrasonic sensor, we need to set two Timers. If you refer back to the specifications of our HC-SR04 Ultrasonic sensor in the table above, you will find that it says “Trigger Signal — at least 10us long “. This means that when we set our Trig pin HIGH, we have to wait for 10 microseconds and then turn it back to LOW. To achieve this 10 microseconds delay, we have to set a Timer

another timer for this. Lets call this TIMER0.

The first thing you should know about Timers is that, they are directly affected by the running frequency of the microcontroller.

Lets say we running on a TM4C123xx Tiva LanchPad which by default runs at 16MHz. To set the microsecond delay Timer, after performing the necessary initialization requirements such as enabling the clock for the Timer block , selecting the bit option and selecting the running mode whether periodic vs. one-shot and down vs. up counter, you will have to load the number 16 minus 1 into the “Timer Interval Load Value Register”

This is how we arrive at 16 :

16MHz literally means 16,000,000 clock cycles per second.

Remember , 1e-6 microseconds = 1 second

1 second = 16,000,000 clock cycles

1 microsecond = 1e-6 microseconds x 16,000,000 cycles

= 16

Therefore at 16 MHz, our microcontroller runs 16 clock cycles in 1 microsecond.

To learn more about Timers please take a look at the lessons on Timers.

This is what the delay_Microsecond function looks like :

(Written in CMSIS standard,Implemented on Texas Instrument TM4C123 TivaC)

```
void delay_Microsecond(uint32_t time)

{

int i;

SYSCTL->RCGCTIMER |=(1U<<1); TIMER1->CTL=0;
```

```
TIMER1->TAMR=0x02;

TIMER1->TAILR= 16-1;

TIMER1->ICR =0x1;

TIMER1->CTL |=0x01;

for(i=0;i<time;i++) { while((TIMER1->RIS & 0x1)==0);

TIMER1->ICR = 0x1;

}

}
```

Now we are in a position to create microsecond delays. To create a 10 microseconds delay as required by the Trig. pin, we simply need to call our delay_Microsecond function like this

delay_Microsecond(10)

Now lets take a look at how we are going to use **TIMER0 to get the rising and falling edges of our Echo pin**. Again, using our circuit diagram above as an example, we can see the the Echo pin is connected pin PB6 of our microcontroller.

In our function for initializing this Timer,we have set PB6 Alternate Function Select Register(AFSEL), set the appropriate PCTL of the pin (this can be found in the datasheet), set the bit option and finally set the running mode as both edges input capture mode. Again for details on the meaning and functions of these terms, please take a look at the lessons on Timers .

This is what the initialization function looks like:

```
void  Timer0_init(void)

{
```

```
GPIOB->DEN |=ECHO;

GPIOB->AFSEL |=ECHO;

GPIOB->PCTL &=~0x0F000000;

GPIOB->PCTL |= 0x07000000;

TIMER0->CTL &=~1;

TIMER0->CFG =4;

TIMER0->TAMR = 0x17;

TIMER0->CTL |=0x0C;

TIMER0->CTL |=1;

}
```

Measuring the distance of an obstacle

To measure the distance of an obstacle in front of the ultrasonic module, we have to:

1. Set the Trig pin LOW , to just make sure its low
2. Wait for about 10 microseconds for it to take effect
3. Set the Trig pin HIGH
4. Wait for at least 10 microseconds
5. Set the Trig pin LOW
6. Start Timer0
7. Capture the time stamp of the rising edge
8. Capture the time stamp of the falling edge
9. Get the difference of the two time stamps

time_difference = 62.5e-9 * 5882

This is what this piece of code looks like :

```
const double _16MHz_1clock = 62.5e-9; /*Value of 1clock cycle in nanoseconds*/

const uint32_t MULTIPLIER = 5882; /*Derived from speed of sound*/

uint32_t measureD(void){

GPIOA->DATA &=~TRIGGER;

delay_Microsecond(12);

GPIOA->DATA |= TRIGGER;

delay_Microsecond(12);

GPIOA->DATA &=~TRIGGER;

/*Capture firstEgde i.e. rising edge*/

TIMER0->ICR =4;

while((TIMER0->RIS &4)==0){}; //Wait till captured

highEdge = TIMER0->TAR;

/*Capture secondEdge i.e. falling edge */

TIMER0->ICR =4; //cleaer timer capture flag

while((TIMER0->RIS &4) ==0){};

lowEdge = TIMER0->AR;

ddistance = lowEdge -highEdge;

ddistance = _16MHz_1clock *(double) MULTIPLIER *(double)ddistance;
```

```
}
```

To view the complete driver code as implemented in a program, click here.

Arduino