1. Zur Ansteuerung des Ultraschallboards nutzen Sie das Trainingsboard XC866 und

erstellen hierfür ein Programm zur Abstandsmessung für Entfernungen von 3 cm bis 3 m.

2. Skizzieren Sie den Ablaufplan der Messaufgabe in einem Struktogramm

3. Zunächst erstellen Sie ein Programm, das eine Entfernungsmessung zwischen 0,1 m und

0,8 m ermöglicht. Eine Timerüberlaufsteuerung ist hierzu noch nicht notwendig. Die zur

Verfügung stehende Timerzeit von T0 (16 Bit) reicht hierzu aus. Die jeweilige Entfernung

soll am LCD des Trainingsboards angezeigt werden.

4. Im zweiten Schritt erweitern Sie die Messentfernung auf 3 m.

5. Der dritte Schritt beinhaltet den Einbezug der sehr kleinen Entfernungen. Zeigen Sie mit

Hilfe des Analog Discovery auf, worin die Schwierigkeiten bei der Messung von sehr

kleinen Entfernungen bestehen. Erweitern Sie ihr Programm auf kleine Entfernungen bis

0,03 m.

1. 3cm bis 3m

Hardware Aufbau:

Die Abstandsmessung erfolgt mit dem Sensor SF04 und dem XC866 Board.

Ein Bild, das Elektronik, Schaltung, Elektronisches Bauteil, Elektrisches Bauelement enthält.

Automatisch generierte Beschreibung

Für die Realisierung eines Abstandsmesssystems wurde der SF04 Ultraschallabstandssensor in Kombination mit dem XC866-Trainingsboard verwendet. Der SF04-Sensor ist ein präziser und zuverlässiger Sensor, der für die Abstandsmessung mittels Ultraschalltechnologie konzipiert ist. Er erzeugt hochfrequente Schallwellen und empfängt das Echo, das von einem Objekt reflektiert wird. Die Zeit zwischen dem Senden des Schallimpulses und dem Empfang des Echos ist proportional zur Entfernung zwischen dem Sensor und dem Objekt.

Das XC866-Trainingsboard ist eine vielseitige Entwicklungsplattform, die speziell für die Implementierung von Mikrocontroller-Anwendungen entwickelt wurde. Es verfügt über eine Vielzahl von Schnittstellen und Anschlüssen, die eine flexible und einfache Integration von Peripheriegeräten wie dem SF04-Sensor ermöglichen.

Der Sensor wird mit dem XC866-Board wie folgt verbunden:

Stromversorgung: Der SF04-Sensor benötigt eine Versorgungsspannung, die typischerweise bei 5V DC liegt. Die roten und schwarzen Drähte in der Abbildung stellen die Spannungsversorgung des Sensors sowie des Boards dar. Diese werden mit 5V versorgt.

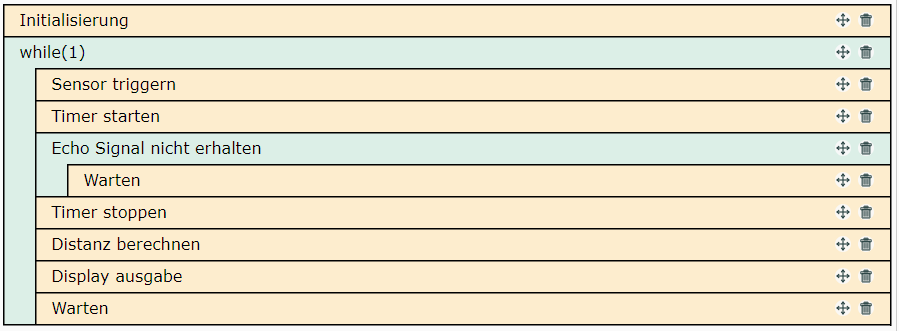
Trigger- und Echo-Pins: Der Trigger-Pin des Sensors, der für das Auslösen der Ultraschallimpulse verantwortlich ist, wird an den dafür vorgesehenen Port-Pin (zum Beispiel P1.2) des XC866-Boards angeschlossen, wie im Code als TRIGGER\_PIN definiert. Der Echo-Pin, der das reflektierte Signal empfängt, wird ebenfalls an den entsprechenden Port-Pin (zum Beispiel P0.5) angeschlossen, der im Code als ECHO\_PIN definiert ist. Der gelbe und der blaue Draht repräsentieren die Signalleitungen für Trigger und Echo. Sie werden an die zuvor erwähnten Pins auf dem XC866-Board angeschlossen, um die Kommunikation zwischen dem Sensor und dem Mikrocontroller zu ermöglichen.

Die korrekte Verdrahtung ist entscheidend, um eine präzise Signalübertragung zu gewährleisten.

Nachdem die physischen Verbindungen hergestellt wurden, erfolgt die Initialisierung der Pins und Schnittstellen im Programmcode. Die Konfiguration der Timer und Interrupts auf dem Mikrocontroller ermöglicht eine genaue Zeitmessung der Ultraschall-Signale, die für die Abstandsberechnung erforderlich ist.

Insgesamt bietet der beschriebene Hardwareaufbau eine solide Basis für eine präzise und zuverlässige Abstandsmessung und illustriert die effektive Nutzung von Mikrocontroller-Ressourcen in Verbindung mit Sensortechnologie.

Umsetzung:



Initialisierung

Die Initialisierung Ihres Systems wird durch die Funktion init() durchgeführt, die zu Beginn des main()-Programms aufgerufen wird. Diese Funktion ist entscheidend, um das System in einen definierten Ausgangszustand zu bringen und umfasst mehrere Schritte:

LCD-Initialisierung: lcd\_init() konfiguriert das LCD-Display, welches für die Ausgabe der Messergebnisse verwendet wird. Mit lcd\_clr() wird das Display anschließend gelöscht, um sicherzustellen, dass keine vorherigen Daten angezeigt werden.

Port-Konfiguration: P0\_DIR und P1\_DIR sind Konfigurationen für die digitale I/O (Input/Output). Der Wert 0x00 konfiguriert alle Pins von Port 0 als Eingänge, und der Wert 0xFF konfiguriert alle Pins von Port 1 als Ausgänge. Diese Konfiguration ist wichtig, da der Trigger- und Echo-Pin entsprechend als Ausgang bzw. Eingang eingestellt werden müssen.

Timer-Modus: TMOD = 0x11; setzt die Timer 0 und 1 in den 16-Bit-Zählmodus. Dies ist wesentlich für die genaue Zeitmessung, die benötigt wird, um die Dauer des Ultraschallsignals zu erfassen.

Timer 1 Start: TR1 = 1; startet Timer 1, der für die wait-Funktion verwendet wird, um eine bestimmte Verzögerungszeit zu erzeugen.

Interrupt-Konfiguration: Der externe Interrupt wird mit EA = 1; global aktiviert und für den externen Interrupt 0 durch EX0 = 1; spezifisch freigeschaltet. IT0 = 1; konfiguriert den Interrupt 0 als Edge Triggered, um auf eine Signaländerung zu reagieren, und EXICON0 = 0x00; stellt ihn so ein, dass er auf eine fallende Flanke (Übergang von High zu Low) des Echosignals reagiert.

Sensor Triggerung

Im Hauptprogramm main() wird nach der Initialisierung in einer Endlosschleife while(1) die Funktion sendTriggerPulse() aufgerufen, um den Messvorgang zu starten.

Setzen des Trigger-Pins: P1\_DATA = (1 << TRIGGER\_PIN); setzt den Trigger-Pin auf High. Durch die Bit-Verschiebung wird sichergestellt, dass nur das Bit, das dem Trigger-Pin zugeordnet ist, gesetzt wird, während alle anderen Bits auf ihrem vorherigen Zustand bleiben.

Wartezeit für Trigger-Impuls: wait(1); ruft eine Verzögerungsfunktion auf, die einen kurzen Trigger-Impuls erzeugt. Die Funktion wait ist so programmiert, dass sie eine Schleife verwendet, um eine genau definierte Verzögerung zu erzeugen, basierend auf der Timer1-Konfiguration. Jede Iteration der Schleife dauert annähernd 15 Mikrosekunden, was durch die Einstellung des Timer1 Registers erreicht wird.

Rücksetzen des Trigger-Pins: P1\_DATA = ~(1 << TRIGGER\_PIN); setzt den Trigger-Pin zurück auf Low und beendet somit den Trigger-Impuls.

Timer Start für Echo-Zeitmessung

Zurücksetzen des Timer-Registers: Die Werte TH0 und TL0 werden auf 0 gesetzt, um den Timer0 zu initialisieren. Dies stellt sicher, dass die Messung der Echo-Laufzeit von einem bekannten Ausgangspunkt startet.

Timer 0 Start: TR0 = 1; schaltet Timer 0 ein, was bedeutet, dass der Timer jetzt läuft und die Zeit zählt, bis das Echo-Signal empfangen wird.

Warten auf das Echo-Signal

Die Schleife while(TR0 == 1); hält die Ausführung des Programms an dieser Stelle an, bis der externe Interrupt ausgelöst wird, was bedeutet, dass das Echo-Signal empfangen wurde.

Echo Interrupt Service Routine

Timer Stopp bei Echo-Empfang: Wenn das Echo-Signal empfangen wird, wird die Interrupt Service Routine echo\_interrupt() ausgelöst, was zur Folge hat, dass TR0 = 0; ausgeführt wird und somit Timer 0 gestoppt wird.

Löschen der Interrupt-Anforderung: IRCON0 = 0; löscht die Interrupt-Anforderung, um sicherzustellen, dass der Mikrocontroller für weitere Interrupts bereit ist.

Das Stoppen des Timers bei Empfang des Echosignals ist entscheidend, da die gemessene Zeit direkt zur Berechnung der Entfernung verwendet wird.

Diese Schritte bilden zusammen den Kern des Messvorgangs und sind entscheidend für die Genauigkeit des Systems. Der Code zeigt, dass die Zeitmessung vom Senden bis zum Empfangen des Echosignals genutzt wird, um die Distanz zu einem Objekt zu bestimmen.

Berechnung der Distanz

Die calculate\_distance()-Funktion dient der Bestimmung des Abstands zu einem Objekt unter Nutzung der Zählwerte des Timers (TH0 und TL0) eines XC886 Mikrocontrollers. Die Schallgeschwindigkeit in Luft beträgt standardmäßig 343 m/s bei Raumtemperatur. Da das Schallsignal zum Zielobjekt und zurück zum Sensor reist, wird die effektive Ausbreitungsgeschwindigkeit für die Distanzberechnung halbiert. Dies entspricht einer Geschwindigkeit von ca. 172 m/s. Der Mikrocontroller arbeitet mit einer Taktfrequenz von 12 MHz. Unter Anwendung dieser Parameter konvertiert die Funktion die Timer-Zählerwerte in eine Distanzangabe, indem sie die Anzahl der Timer-Ticks durch die Frequenz des Mikrocontrollers teilt, um die Zeit zu ermitteln, die der Schall für die zurückgelegte Strecke benötigt hat. Anschließend wird diese Zeit mit der halbierten Schallgeschwindigkeit multipliziert, um die Distanz in Metern zu berechnen. Die Berechnung passt die Einheiten entsprechend an, um das Ergebnis in Zentimetern zu erhalten.

Formel:

Einheiten anpassen

XC886 arbeitet mit 12MHz dementsprechend:

**Erweiterung des Messbereichs und Einbeziehung kleiner Entfernungen**

Ein Bild, das Text, Screenshot, Reihe, Zahl enthält.

Automatisch generierte BeschreibungIn der Weiterentwicklung des Abstandsmesssystems wurde die Funktionalität des Codes dahingehend erweitert, dass nun auch größere Entfernungen bis zu 3 Metern erfasst werden können. Zudem wurde die Messgenauigkeit für kleine Entfernungen bis hinab zu 3 Zentimetern verbessert. Diese Anpassungen sind essentiell, um das Anwendungsspektrum des Systems zu vergrößern und die Präzision der Messungen zu optimieren.

Timer Überlauf Handhabung

Um die Erfassung größerer Entfernungen zu ermöglichen, wurde der Code um eine Interrupt Service Routine (ISR) für den Timer-Überlauf ergänzt. Die timer\_ueberlauf()-Routine wird aktiviert, sobald der Timer 0 sein Zählmaximum überschreitet. Innerhalb dieser Routine wird der Variable t0\_ueberlauf jeweils der Wert 65536 hinzugefügt, welcher der maximalen Zählmenge des 16-Bit-Timers entspricht. Diese Maßnahme erlaubt es, die Zeitspanne für längere Distanzen zuverlässig zu messen, da die Zeit zwischen den Trigger- und Echo-Ereignissen den Timer mehrmals überschreiten lassen kann.

Fazit

Durch die Integration dieser Funktionen in den Code konnte das Messsystem erfolgreich auf einen erweiterten Bereich von 3 Zentimetern bis zu 3 Metern ausgebaut werden. Die sorgfältige Implementierung der Timer-Überlaufbehandlung und die präzise Berechnung der Entfernungen gewährleisten eine hohe Genauigkeit der Messergebnisse. Die benutzerorientierte Anzeige der Messwerte auf dem LCD erhöht zudem die Benutzerfreundlichkeit des Systems.

Problematik der Messung von geringen Distanzen mittels Ultraschallsensoren

Bei der Konzeption des Abstandsmesssystems, das auf der Ultraschalltechnologie basiert, ergibt sich eine inhärente Schwierigkeit bei der Erfassung von sehr geringen Distanzen. Der implementierte SF04 Ultraschallsensor ist aufgrund seiner Bauweise und Funktionsprinzipien nicht in der Lage, Objekte zu detektieren, die näher als 3 Zentimeter am Sensor platziert sind. Dieses Limit ist durch die physischen Eigenschaften der Schallausbreitung und die konstruktive Gestaltung des Sensors bedingt.

Ultraschallsensoren messen Distanzen durch das Aussenden von Schallwellen und das Empfangen des Echos dieser Wellen nach der Reflexion an einem Objekt. Bei sehr kurzen Distanzen kann es vorkommen, dass das Echo des ausgesendeten Signals nicht zurück zum Empfängerteil des Sensors gelangt. Ein Grund hierfür ist, dass bei einer derartigen Nähe des Objekts zum Sensor die Schallwellen direkt absorbiert oder seitlich wegreflektiert werden können, ohne dass sie den Empfänger erreichen. Wenn beispielsweise ein Finger unmittelbar vor dem Sensor platziert wird, kann dies dazu führen, dass der Sensor kein Echo empfängt und daher fälschlicherweise annimmt, dass kein Objekt vorhanden ist.

Diese Begrenzung ist nicht nur auf die physische Blockierung des Echosignals zurückzuführen, sondern auch auf die zeitliche Überlappung von Sende- und Empfangssignal bei sehr kurzen Distanzen. Die Zeit, die das Signal benötigt, um zum Objekt und zurück zum Sensor zu gelangen, kann so kurz sein, dass das Echo noch während der Dauer des ausgesendeten Impulses ankommt. In solch einem Fall ist es für den Sensor technisch schwierig, das Echo vom ursprünglichen Signal zu unterscheiden.

Messfehler

|  |  |  |  |
| --- | --- | --- | --- |
| **Abstand Zollstock** | **Abstand Messung** | **Abweichung abs** | **Abweichung rel** |
| 3 | 3 | 0 | 0% |
| 10 | 10 | 0 | 0% |
| 20 | 20 | 0 | 0% |
| 30 | 30 | 0 | 0% |
| 40 | 40 | 0 | 0% |
| 50 | 50 | 0 | 0% |
| 60 | 60 | 0 | 0% |
| 70 | 69 | 1 | 1% |
| 80 | 79 | 1 | 1% |
| 90 | 90 | 0 | 0% |
| 100 | 99 | 1 | 1% |
| 120 | 119 | 1 | 1% |
| 140 | 139 | 1 | 1% |
| 160 | 159 | 1 | 1% |
| 180 | 179 | 1 | 1% |
| 200 | 199 | 1 | 1% |
| 250 | 249 | 1 | 0% |
| 300 | 298 | 2 | 1% |

Flags:

TH0 wird beim Überlauf auf 0 gesetzt

TL0 wird beim Überlauf auf 0 gesetzt

IRCON0 Wird ein externer Interrupt erkannt, so muss das entsprechende Flag im IRCONx-Register in der ISR gelöscht werden (z.B. Bit 0 beim externen Int. 0 in IRCON0).

EA Globale Freigabe,

EX0 External Interrupt 0

ET0 Timer0 Interrupt

IT0 = 1; // Eingang flanken- oder pegelsensitiv, hier Flanke

EXICON0

Ein Bild, das Text, Schrift, Quittung, Zahl enthält.

Automatisch generierte Beschreibung

Frage – Was ist IE0 ???