Bernburg
Dessau
Köthen

Hochschule Anhalt
Anhalt University of Applied Sciences

emw
Fachbereich
Elektrotechnik, Maschinenbau
und Wirtschaftsingenieurwesen

Hausarbeit

zum

Berufspraktikum

Yang Zhang

Vorname Nachname

Elektro-und Informationstechnik, 2015, 4062576

Studiengang, Matrikel, Matrikelnummer

Jinlin Xie

Vorname Nachname

Elektro-und Informationstechnik, 2015, 4062863

Studiengang, Matrikel, Matrikelnummer

Thema:

Konzeption und Entwicklung eines universellen Alarmsystems mit einem Beaglebone Black und angeschlossenen Sensoren

Prof. Dr. Ingo Chmielewski

Hochschulmentor(in)

08. 07. 2019 – 13. 11. 2019

Berufspraktikum vom – bis

13.11.2019

Inhaltsverzeichnis

1 Einleitung

Sicherheit in Innenräumen ist heutzutage sehr notwendig und spielt eine immer wichtigere Rolle. Mit der modernen Technologie werden Wohnungen oder Büro ziemlich bequemer und angenehmer gemacht. Bei dem Projekt handelt es sich auch um Sicherheit in Innenräumen. Das Ziel des Projektes ist die Innenüberwachung, welche durch den Beaglebone black, IR-Sensor, PIR-Sensor und Magnetsensor realisiert wird. Wenn ein Sensor detektiert, soll eine Nachricht über das Alarmsystem überwacht werden.
Dieses System besteht hauptsächlich aus zwei* Teilen. Der erste Teil beschreibt die theoretischen Grundlagen und der zweite Teil erläutert die praktische Ausführung. All dies wird in den kommenden Kapiteln ausführlich erläutert.

2 Theoretische Grundlagen

Dieses Kapitel erläutert das GPIO-Bussystem, das PWM-Bussystem. Die Eigenschaften und Wirkungsweisen der beiden Bussysteme. Die verwendeten Module von Python werden ausführlich erklärt, damit man die praktische Ausführung besser verstehen kann.

2.1 PWM-Bussystem

PWM ist eine Art von digitaler Modulation, bei der technologischen Größe (z. B. die Spannung) zwischen zwei Werten variiert. Bei konstanter Frequenz wird ein Rechteckimpuls moduliert, dessen Breite, Breite oder Länge sich ändert. Die Beziehung zwischen Impulsen und Pausen wird als Arbeitszyklus bezeichnet.

Bei der Pulsdauermodulation hat das modulierte Signal eine feste Amplitude. Zu diesem Zweck hängt die Pulsdauer von der Amplitude des Informationssignals ab. Je stärker das Informationssignal ist, desto länger ist die Pulsdauer. Je negativer das Informationssignal, desto kürzer ist der Impuls.

Anwendung der PWM

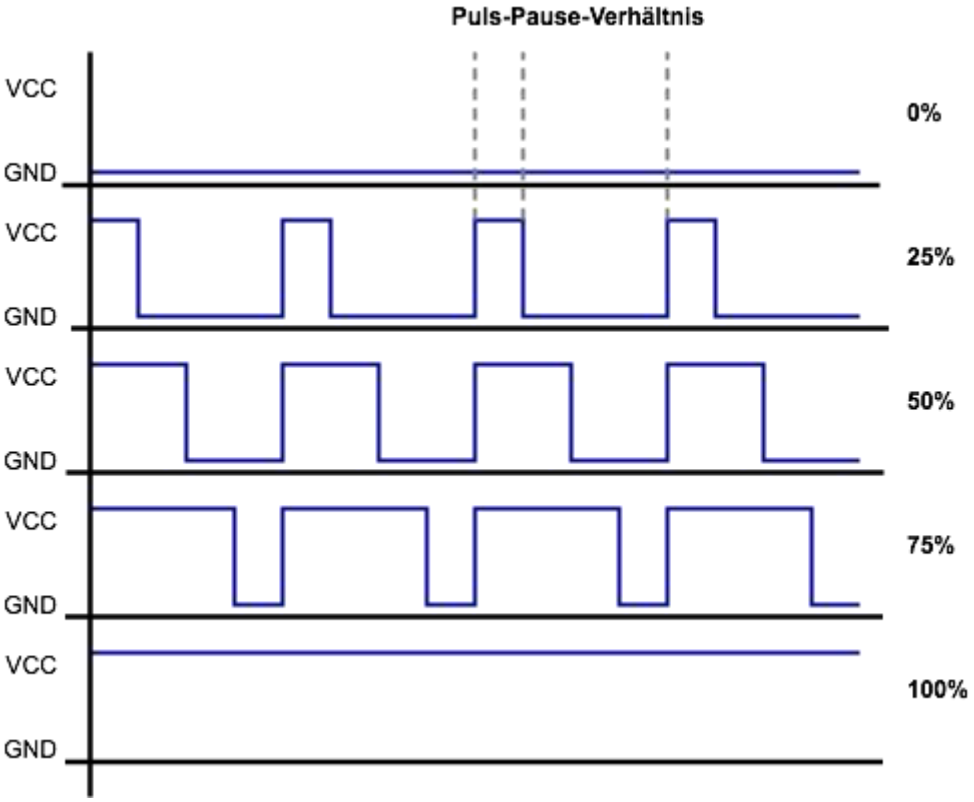


Abbildung 1: Grundstruktur eines PWM-Bussystems

Obwohl das Signal der Pulsweitenmodulation eine Wechselspannung oder insbesondere eine gemischte Spannung ist, kann es verwendet werden, um die Leistung einer Gleichstromlast (z. B. LED, Motor, Heizwiderstand usw.) zu regeln. Anstatt diese Komponenten durch die Höhe der Betriebsspannung zu steuern, wird die Spannung oder der Strom durch Pulsweitenmodulation einfach in kurzer Zeit unterbrochen. Dies stellt eine Beziehung zwischen dem Spannungsimpuls und der Pause her. Dieses Verhältnis entscheidet über die effektive Spannung.

Beispiel: Wenn das Verhältnis zwischen Impuls und Pause gleich 50% ist, ergibt sich eine effektive Spannung von 6 V und eine Impulsspannung von 12.

2.2 GPIO-Bussystem

GPIO ist eine Abkürzung für General Purpose Input und Output. Hiermit werden programmierbare Ein- und Ausgänge für den allgemeinen Gebrauch festgelegt. GPIOs werden als Lötstifte oder -stifte angeheftet oder herausgesteckt und als Schnittstelle zu anderen Systemen oder Schaltkreisen verwendet, um wir über den Beaglebone Black zu steuern. Nach dem Programmieren kann der Beaglebone Black digitale Signale (Eingänge) von außen annehmen oder Signale nach außen senden (Ausgang).

Viele GPIOs haben je nach Einrichtung und Programmierung unterschiedliche Funktionen. Zusätzlich zu den typischen GPIO-Ein- und Ausgängen gibt es auch Pins mit den Doppelfunktionen I2C, SPI und serielle Schnittstelle.

Die GPIOs sind oftmals nur als Lötpunkte, beim Beaglebone Black vorteilhafterweise als Stiftleiste, herausgeführt. Viele der Stifte sind flexibel und einige sind sogar doppelt belegt. Das heißt, wir müssen ihnen durch Programmierung Funktionen zuweisen, bevor wir verwenden können. In jedem Fall müssen wir die Funktionalität

initialisieren, bevor wir GPIO verwenden können. Schließt mögliche führende Stifte aus. Ihre Spannung ist immer fest.

GPIO-Pins weisen eine Vielzahl elektrischer Eigenschaften auf, die für Neulinge in der Elektronik verwirrend sein können. Im Allgemeinen müssen jedoch alle elektrischen Systeme elektrische Eigenschaften berücksichtigen, da es sich häufig um Parameter und Grenzwerte handelt, die die Funktion und den Betrieb des elektrischen Systems bestimmen. Kurz gesagt, das Beobachten der elektrischen Eigenschaften jeder elektronischen Komponente bestimmt, ob die Schaltung richtig funktioniert. Oder das System ist nach dem Debuggen beschädigt und es liegt ein Fehler vor. Grundsätzlich wird der Beaglebone Black mit 5 Volt betrieben. Das System arbeitet jedoch mit einem 3,3-Volt-Chip, was bedeutet, dass auch GPIOs mit 3,3 Volt arbeiten können.

3 Praktische Ausführung

In diesem Kapitel werden die einzelnen, konkreten Schritte beim Verbinden von Erweiterungsplatinen mit dem Beaglebone Black und das Wirkungsprinzip des Überwachungsprogramms ausführlich erklärt.

3.1 Beschreibung des Projekts

Das Ziel des Projektes ist die Detektion und die Signalisierung der Alarmanlage mit dem Beaglebone Black.
Die Durchführung besteht hauptsächlich aus zwei Schritten:
Der erste Schritt ist die Verbindung der geeigneten Komponenten mit dem Beaglebone Black.
Der zweite Schritt ist die Entwicklung eines geeigneten Programms für die Überwachung der Komponenten.

3.2 Verwendete Komponenten

3.2.1 Beaglebone Black

BeagleBone Black ist eine kostengünstige, von der Community unterstützte Entwicklungsplattform für Entwickler und Bastler. Booten wir Linux in weniger als 10 Sekunden und starten wir die Entwicklung in weniger als 5 Minuten mit nur einem USB-Kabel.

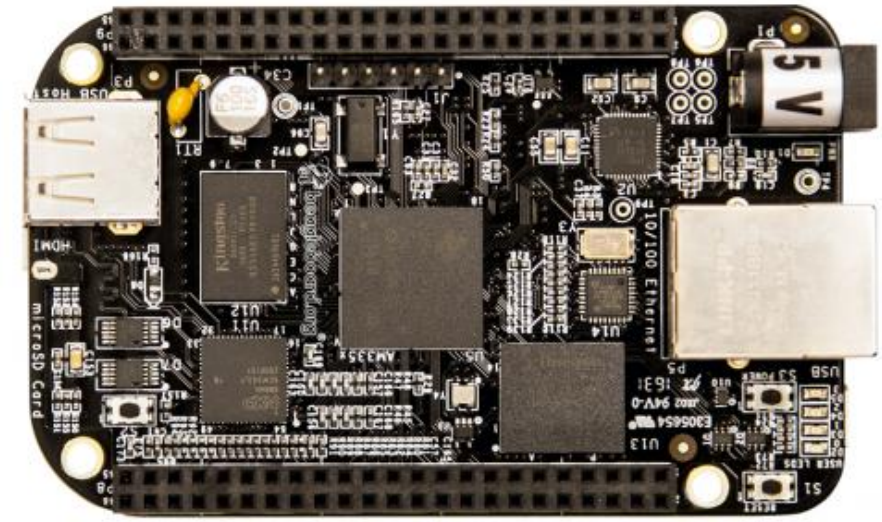


Abbildung 2: Draufsicht von Beaglebone Black

Beaglebone Black bietet die neueste Version von 512 MB RAM (LP DDR3), einem Cortex-A8 des 1-GHz-Sitara-AM3358-ARM von Texas Instruments, im Gegensatz zu seinem Vorgänger, dem Beagle Board der Beagle Board-Kompaktversion des Hardware-Hackers, einen integrierten Micro-HDMI-Anschluss und 4 GB eMMC-Flash. Vor Ort sofort die Platte debuggen können Ångström-Linux montiert eMMC. Das

System unterstützt Android, Ubuntu, Debian, openSUSE und Ångström sowie verschiedene andere Betriebssysteme wie FreeBSD, OpenBSD, QNX und Windows Embedded.

Für BeagleBone kann eine Vielzahl von Erweiterungskarten (Cape) verwendet werden, wodurch die Funktionalität erheblich erweitert wird. An den zwei Reihen von 46-poligen Erweiterungsanschlüssen des BeagleBone ist ein Umhang angebracht, der wiederum entsprechende Schnittstellen für die Erweiterung mit anderen Umhängen bietet. Ein BeagleBone kann bis zu vier Umhänge gleichzeitig erweitern.

3.2.2 PIR-Sensor

PIR-Sensor ermöglicht es Ihnen, zu erkennen, Bewegung, fast immer verwendet, um festzustellen, ob die Menschheit Sensorbereich eingetreten ist. wir sind klein, kostengünstig, stromsparend, einfach zu bedienen und unterliegen keinem Verschleiß. Daher sind wir in der Regel in Haushaltsgeräten oder in Unternehmen zu finden. wir werden häufig als PIR-, "Passiv-Infrarot" -, "thermoelektrische" - oder "IR-Bewegung" -Sensoren bezeichnet.

Der PIR-Sensor selbst verfügt über zwei Steckplätze, die jeweils aus einem IR-empfindlichen Spezialmaterial bestehen. Das hier verwendete Objektiv macht nicht viel, so dass wir sehen, dass die beiden Schlitze einen bestimmten Abstand "sehen" können (im Grunde genommen die Empfindlichkeit des Sensors). Wenn sich der Sensor im Leerlauf befindet, erfassen beide Steckplätze die gleiche Menge an IR, die Menge an Umgebungsstrahlung aus dem Raum oder der Wand oder von außen.

Wenn ein warmes Objekt wie ein Mensch oder ein Tier vorbeikommt, fängt es zuerst die Hälfte des PIR-Sensors ab, was zu einer positiven Differenzänderung zwischen den beiden Hälften führt. Wenn der warme Körper den Erfassungsbereich verlässt, ändert sich die Situation umgekehrt, wodurch der Sensor eine negative Differenzänderung erzeugt. Diese Änderungsimpulse werden erkannt.

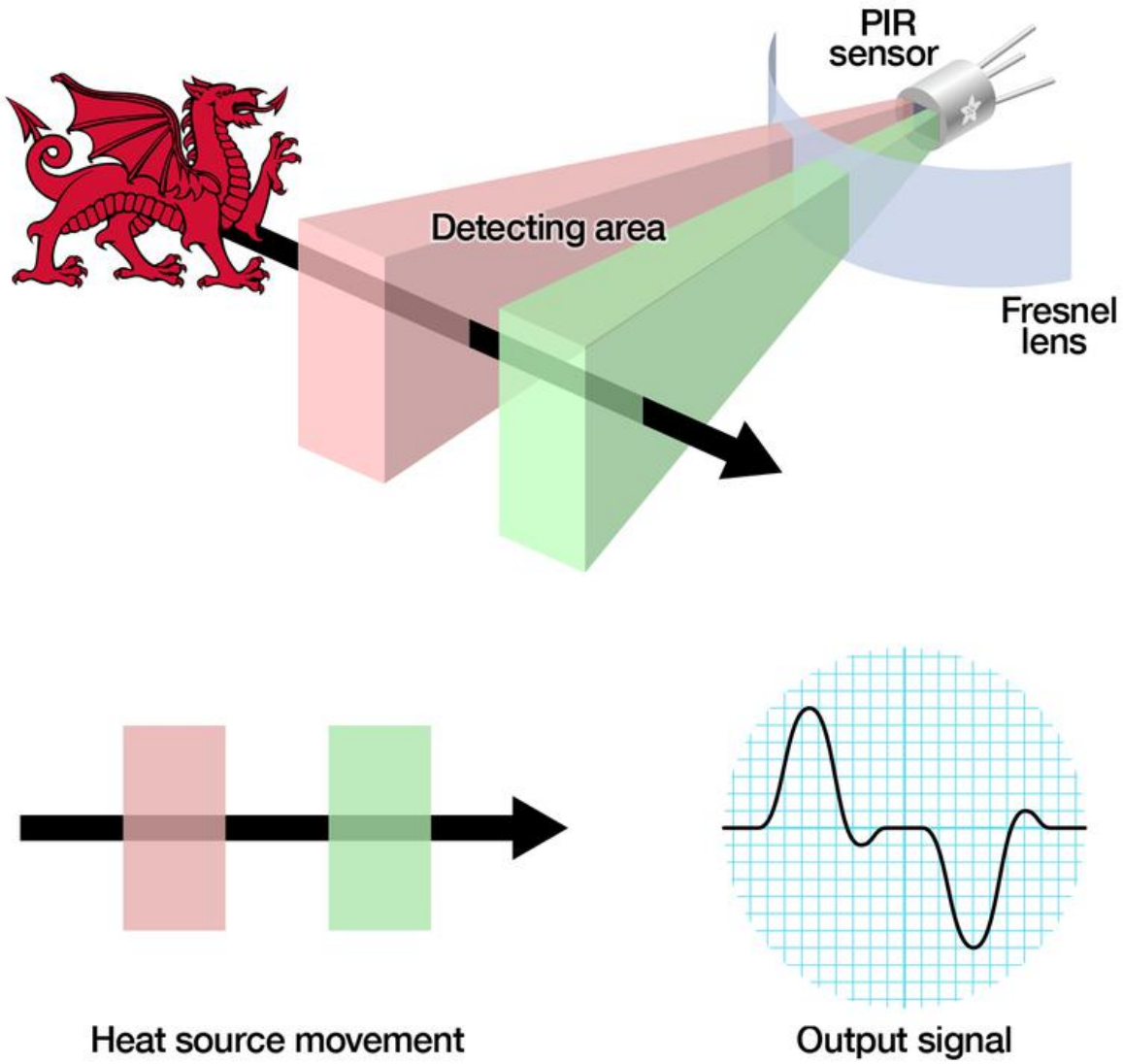


Abbildung 3: Prinzip des PIR-Sensors

3.2.3 IR-Sensor

Bietet einen hochempfindlichen ITR9909-Lichtreflektor, mit dem das Vorhandensein von Objekten vor dem Grove-IR Distance Interrupter v1.2 erkannt werden kann. Der Fotoreflektor besteht aus einer GaAs-Infrarot-Leuchtdiode und einem planaren Silizium-Fototransistor. Wenn sich das Objekt direkt vor dem Sensor befindet, wird das vom Sender emittierte reflektierte Infrarotlicht vom Fototransistor aufgenommen und je nach Abstand zwischen der reflektierenden Oberfläche und dem Sensor logisch hoch oder logisch niedrig ausgegeben.

Handelt es sich bei der reflektierenden Oberfläche um ein weißes Blatt Papier, kann die Erkennungsschwelle durch Einstellen des Potentiometers im Bereich von ca. 90 mm bis 300 mm eingestellt werden. Die Anzeige-LED dient zur Visualisierung der Messung, da wir rot leuchten, wenn ein Objekt erkannt wird, und erlischt, wenn kein Objekt vorhanden ist.

Eigenschaften:

- Einfach zu bedienen
- Integrierter Indikator
- Digitaler Ausgang
- Einstellbarer Erfassungsbereich

3.2.4 Magnet Sensor (Reedschalter)

Reedschalter reagieren auf Magnetfelder. Die Kontakte werden durch einen Magneten (Magnetfeld) geschlossen. Beim Reedschalter schmelzen die Schalterkontakte im Glasrohr und verhindern den Kontakt.

Der Reedschalter besteht aus zwei ferromagnetischen Teilen (normalerweise Nickel / Eisen-Legierung), die in einem Glasrohr hermetisch abgedichtet sind. Die beiden Registerkarten überlappen sich. Wenn das Magnetfeld auf den jeweiligen Schalter wirkt, wobei die beiden Paddel relativ zueinander, ist der Schalter geschlossen. Die Kontaktflächen der beiden Schaltsblätte sind mit einem sehr harten Metall beschichtet, üblicherweise bestehend aus Rhodium oder Ruthenium. Geeignet ist auch Wolfram, Tantal oder ähnlich strukturierte Metalle. Diese können durch Elektroplattieren oder Sputtern (aus der Halbleiterindustrie bekannt) aufgebracht werden. Die Kontaktfläche dieser Hartbeschichtungen garantiert die lange Lebensdauer des Reedschalters. Die vorhandene Luft wird vor dem Schmelzen abgesaugt. Dies geschieht durch Unterdruck. Während des Schmelzprozesses füllen wir den Schalter mit einem Stickstoff- oder Inertgasgemisch mit hohem Stickstoffgehalt.

Abbildung 4: Prinzip des Reedschalters

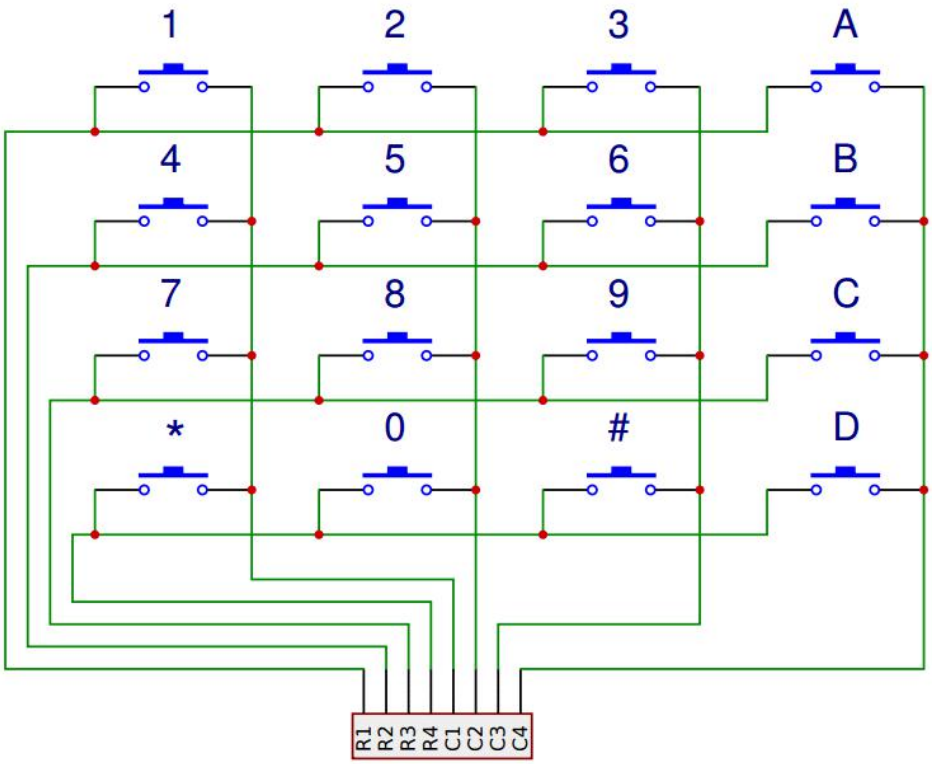
Das vom Permanentmagneten oder der Spule erzeugte Magnetfeld weist auf die entgegengesetzte Polarität, wodurch das Paddel angezogen wird.

·Wenn die Magnetkraft großer als die elastische Kraft des Paddels ist, werden die beiden Kontakte geschlossen.

·Wenn weniger als die Magnetkraft des Schalters ist, öffnet sich der Reed-Schalter wieder.

3.2.5 Tastenfeld

Um die 16 Pins auf dem Beaglebone Black nicht zum Lesen der Tasten der 16 Tastenschalter zu verwenden, werden hier nur 8 Pins zum Multiplexen verwendet. Die acht Anschlüsse der Folientastatur sind mit den acht digitalen Pins des Beaglebone Black verbunden. Vier Pins werden als Ausgang des Beaglebone Black verwendet und mit den vier Leitungen der Tastatur verbunden. Die anderen 4 Pins werden als Eingänge verwendet und mit der Spalte verbunden. An den jeweiligen Kreuzungen liegt ein Membranschalter, der im gedrückten Zustand die Verbindung zwischen Zeile und Spalte



herstellt.

Abbildung 5: Grundschaltung von 4x4 Matrix Tastenfeld

Um Tastatureingaben abzufragen, anstatt die gesamte Tastatur gleichzeitig zu überwachen, wird jede Leitung in kurzer Zeit mit Strom versorgt, das heißt, der entsprechende Ausgangs-Pin auf dem Beaglebone Black geht hoch. Wenn in dieser

Zeit ein Knopf in der Reihe gedrückt wird, wird der Membranschalter ausgeschaltet und HIGH wird am zugehörigen Eingangspin des Beaglebone Black (mit der Säule verbunden) gemessen. Da wir jetzt wissen, welche Leitung Strom liefert und welche Spalte "hoch" ist, müssen wir den Schalter am Schnittpunkt der Spalte und der Spalte drücken. All dies geschieht in Millisekunden, sodass jeder Tastendruck erkannt wird.

Dafür kann man z.B. die Library Keypad verwenden.

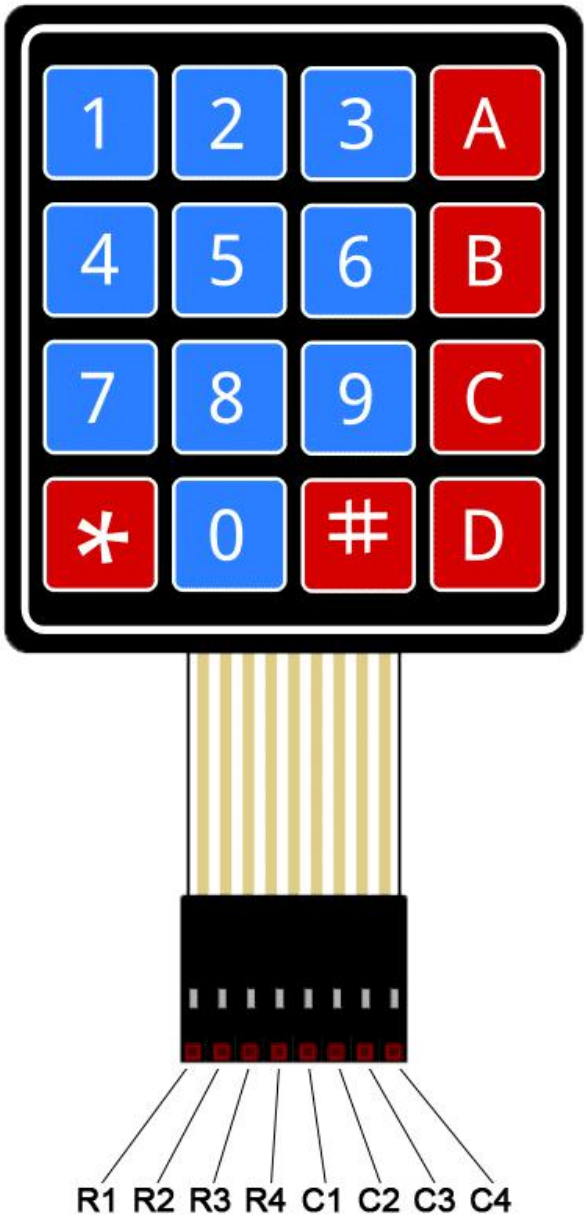


Abbildung 6: Ansicht von 4x4 Matrix Tastenfeld

3.2.6 andere Komponenten

·Lautsprecher

·LEDs

Die beide Komponenten sind Lautsprecher und LEDs ,die in folgenden Verbindungen vernutzte werden.

3.3 Beaglebone Black und die Peripherie

3.3.1 Vorbereitung der Beaglebone Black

Schritt 1:

Laden wir das neueste Debian-Image herunter

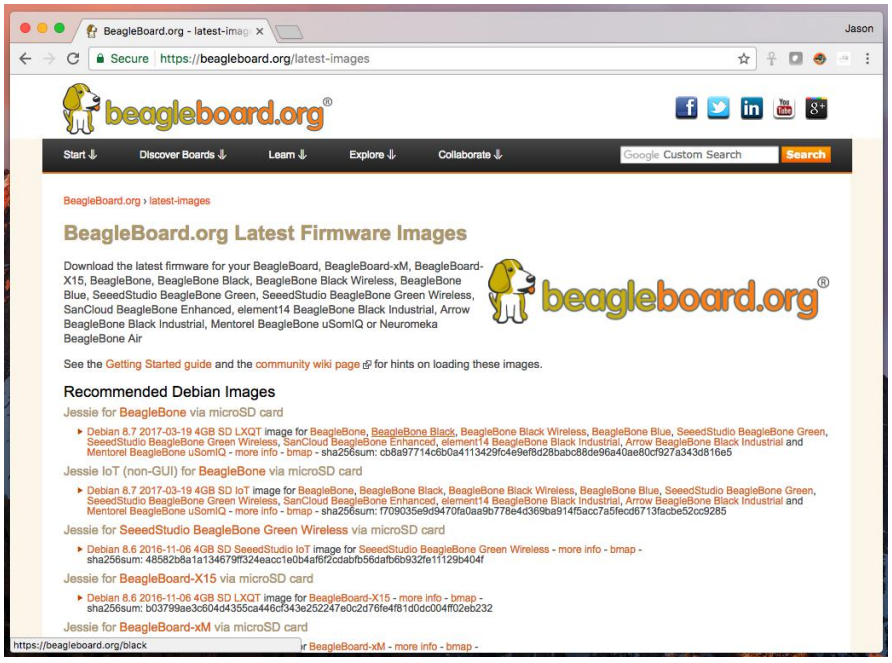


Abbildung 7: Ansicht der Link für Image

Laden wir balenaEtcher herunter und installieren wir es. Schließen wir die SD-Karte über den SD-Steckplatz oder den USB-Adapter Ihres Computers an Ihren Computer an. Verwenden wir den Ätzer, um das Bild auf die SD-Karte zu schreiben. Etcher dekomprimiert das Bild transparent in Echtzeit und schreibt es dann auf die SD-Karte. Eine neu programmierte SD-Karte wird angezeigt. Legen wir die SD-Karte in die (Ausschalt-) Platine ein, drücken und halten wir die USER / BOOT-Taste (wenn Schwarz verwendet wird) und schalten wir über das USB-Kabel oder den 5-V-Adapter ein. Wenn das ursprüngliche BeagleBone oder PocketBeagle verwendet wird, ist der Vorgang abgeschlossen.

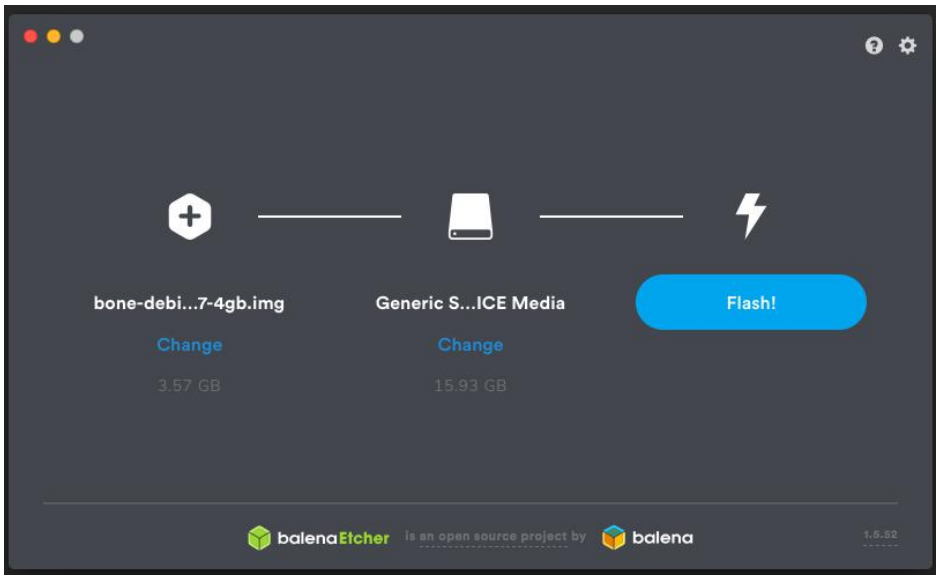


Abbildung 8: Ansicht des Vorgangs zur Brennung von Image auf SD-Karte

Schritt 2:

Wenn wir einen Beagle mit einer SD-Karte (microSD) verwenden, müssen wir diese einlegen, bevor wir das Gerät mit Strom versorgen. Die meisten Beagles verfügen über einen programmierten Onboard-Flash, sodass keine SD-Karte eingelegt werden muss. Wir sehen, dass die Netzanzeige (PWR oder ON) konstant leuchtet. Nach etwa einer Minute sollten andere LEDs in unser Standardkonfiguration blinken.

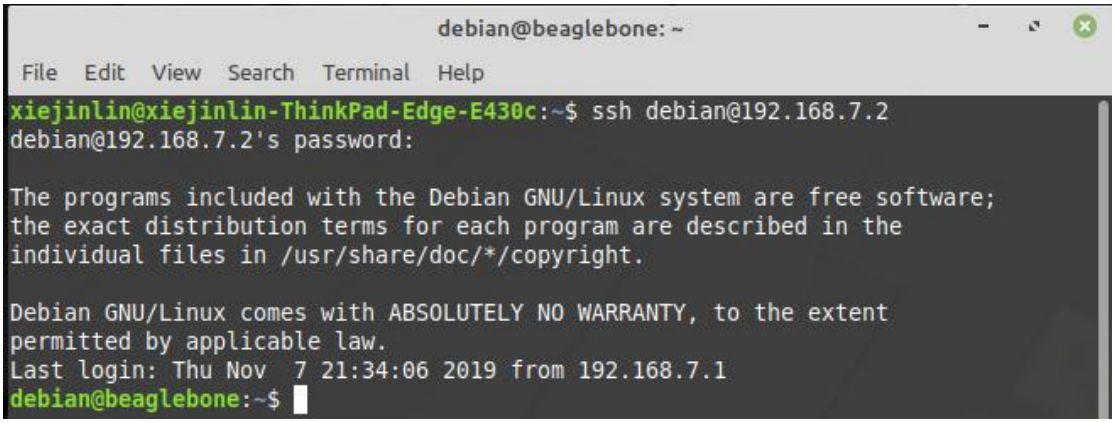


Abbildung 9: Ansicht der erfolgreichen Anmeldung

Bei einer USB-Verbindung sollte der Netzwerkadapter auf unserem Computer angezeigt werden. Auf unserem Beagle sollte ein DHCP-Server ausgeführt werden, der unserem Computer eine IP-Adresse von 192.168.7.1 zur Verfügung stellt, abhängig vom Typ des vom Betriebssystem unseres Computers unterstützten USB-Netzwerkadapters. Unser Beagle reserviert 192.168.7.2 für sich.

3.3.2 Installation der Bibliothek für Beaglebone Black

Adafruit BBIO ist eine API, mit der GPIO-, PWM-, ADC-, UART-, SPI- und eQEP-Hardwarezugriff (Quadrature Encoder) von Python-Anwendungen aus möglich ist, die auf dem Beaglebone Black ausgeführt werden.

Um BBIO installieren zu können, sind folgende Schritte erforderlich.

```
sudo ntpdate pool.ntp.org
sudo apt-get update
sudo apt-get install build-essential python-dev python-pip -y
git clone git://github.com/adafruit/adafruit-beaglebone-io-python.git
cd adafruit-beaglebone-io-python
sudo python setup.py install
```

Abbildung 10: Vorschritt der Installation der Bibliothek

Danach ist die Installation der Adafruit BBIO erfolgreich.

```
debian@beaglebone:~$ pip list
DEPRECATION: The default format will switch to columns
no (list) section) to disable this warning.
Adafruit-BBIO (1.0.10)
```

Abbildung 11: Erfolgreiche installierte Bibliothek

3.3.3 System des Lautsprechers

Zur Verfügung des Lautsprechers nutzen wir ein PWM als Kontroller und MOSFET als Versicherung, um Defekt zu vermeiden.

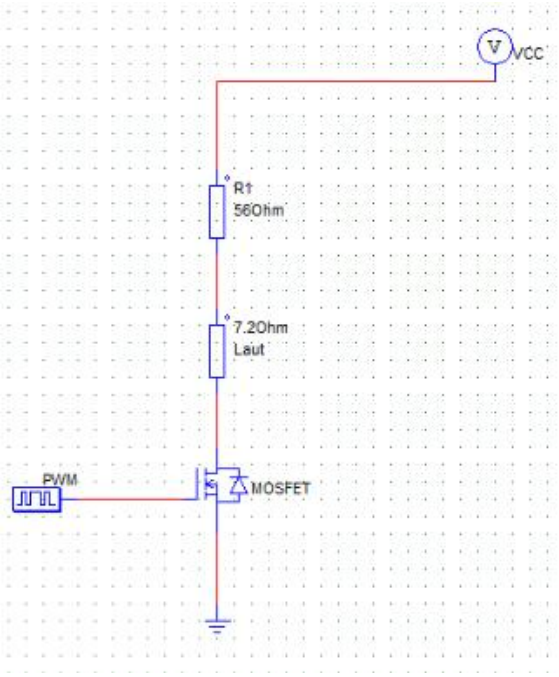


Abbildung 12: Schaltbild des Lautsprechers

Von dem Bild sehen wir eine Schaltung, die aus solchen Komponenten besteht. PWM entsteht eine hochfrequenz laufende Spannung. Durch die Ansteuerung eines

MOSFETs erfolgt über eine Steuerspannung (Gate-Source-Spannung), kann der Stromfluss von Drain nach Source beeinflusst werden.

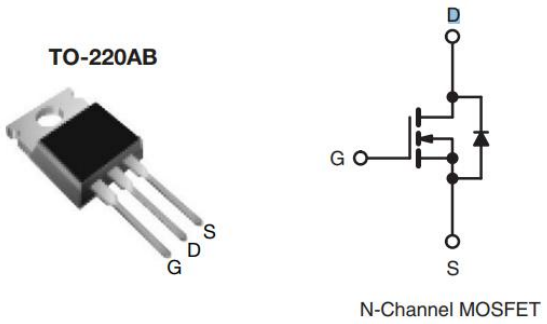


Abbildung 13: Ansicht und Grundstruktur des MOSFETs

Weil die Spannung von VCC die Nennspannung von Lautsprecher überschrittet hat, sollen wir im Drain Teil einen Vorwiederstand zur Verteilung der Spannung verbinden, sodass der Lautsprecher funktionieren kann.

3.3.4 Verbindung andere Komponenten

Mit IR, PIR und Magnet Sensoren verbinden wir auf VCC, GND und Kontakte Pins in Beaglebone Black. Unser 4x3 Matrix Tastenfeld soll mit 7 Kontakte Pins verbindet werden.

3.4 Programmierung

Die Programmierung ist wichtigster Teil des Projekts. Dieses komplette Programm bestehen aus 5 kleine Programme, die für IR, PIR, Magnet Sensor, Lautsprecher und Tastenfeld sind.

Im Folgenden sind die definierten Funktionen in diesem Programm aufgezeigt. Wegen der Beschränkung der Textlänge werden nur die wichtigen Funktionen des Programms dargestellt. Es wird ausführlich erläutert, welchem Zweck die einzelnen Funktionen dienen.

3.4.1 Programm für IR-Sensor

In IR Klasse definieren wir zuerst allen benötigten Variablen, dann wir weisen die Variablen dem entsprechenden Modul zu. Wenn das geeignete Signal detektiert wird, leuchtet 2Hz LED und ertönt der Lautsprecher mit 2Hz vier Sekunden lang.

```
class IR:
    def __init__(self, pin):          # einrichten IR Sensor Element(pin, mode,callback function, Verzögerungszeit)
        self.pin = pin
        GPIO.setup(self.pin, GPIO.IN)
        GPIO.add_event_detect(self.pin, GPIO.RISING, callback=self.cb_ir, bouncetime=50)

    def __del__(self): #Elemente lösen
        print("IR {} deleted.".format(self.pin))
        GPIO.remove_event_detect(self.pin)

    def cb_ir(self, pin): #in globale Variable, wenn anderer Sensor schon detektiert hat, start dieser Sensor nicht
        global alarm
        print("cb_ir called.")
        if not alarm.triggered:
            alarm.trigger()
```

Abbildung 14: Programm von IR-Sensor

3.4.2 Programm für PIR-Sensor

In PIR Klasse definieren wir auch zuerst allen benötigten Variablen, dann wir weisen die Variablen dem entsprechenden Modul zu. Wenn das geeignete Signal detektiert wird, leuchtet 2Hz LED und ertönt den Lautsprecher mit 2Hz vier Sekunden lang.

```
class PIR:      # Wie IR Sensor
    def __init__(self, pin):
        self.pin = pin
        GPIO.setup(self.pin, GPIO.IN)
        GPIO.add_event_detect(self.pin, GPIO.RISING, callback=self.cb_pir, bouncetime=50)

    def __del__(self):
        print("PIR {} deleted.".format(self.pin))
        GPIO.remove_event_detect(self.pin)

    def cb_pir(self, pin):
        global alarm
        print("cb_pir called")
        if not alarm.triggered:
            alarm.trigger()
```

Abbildung 15: Programm von PIR-Sensor

3.4.3 Programm für Magnet Sensor

In Magnet Sensor Klasse definieren wir auch zuerst allen benötigten Variablen, dann wir ordnen die Variablen dem entsprechenden Modul zu. Das Fenster ist zu Beginn geschlossen. Wenn das geeignete Signal detektiert wird, leuchtet 2Hz LED und ertönt der Lautsprecher mit 2Hz vier Sekunden lang. Im Gegenteil arbeiten LED und Lautsprecher nicht.

```
class MAG:   # Zwei Zustand fenster zu und an, Anfangswert sollte mit Zu sein
    def __init__(self, pin):
        self.pin = pin
        self.window_closed = True
        GPIO.setup(self.pin, GPIO.IN)
        GPIO.add_event_detect(self.pin, GPIO.BOTH, callback=self.cb_mag, bouncetime=50)

    def __del__(self): # delete Elemente
        print("MAG {} deleted.".format(self.pin))
        GPIO.remove_event_detect(self.pin)

    def cb_mag(self, pin):
        global alarm
        if self.window_closed:
            print("Fenster geoeffnet...")
            self.window_closed = False
            if not alarm.triggered:
                alarm.trigger()
        else:
            print("Fenster zu...")
            self.window_closed = True
```

Abbildung 16: Programm von Magnet Sensor

3.4.4 Programm für LED und Lautsprecher

LED und Lautsprecher sind Ausgabegeräte des Projekts. Wenn das geeignete Signal von IR, PIR und Magnet Sensor detektiert wird, leuchtet 2Hz LED und ertönt der Lautsprecher mit 2Hz vier Sekunden lang.

```
class ROT(): # ähnlich wie Lautsprecher
    def __init__(self, pin, pwm_perc=50, pwm_freq=1):
        self.pin = pin
        self.pwm_perc = pwm_perc
        self.pwm_freq = pwm_freq

    def __del__(self):
        print("ROT {} deleted.".format(self.pin))

    def start(self):
        PWM.start(self.pin, self.pwm_perc, self.pwm_freq)

    def stop(self):
        PWM.stop(self.pin)
```

Abbildung 17: Programm von LED

```
class LAUT:   # Variable
    def __init__(self, pin, pwm_perc=50, pwm_freq=2000):
        self.pin = pin
        self.pwm_perc = pwm_perc
        self.pwm_freq = pwm_freq
        GPIO.setup(self.pin, GPIO.OUT)

    def __del__(self):
        print("PWM {} deleted.".format(self.pin))

    def start(self):
        PWM.start(self.pin, self.pwm_perc, self.pwm_freq)

    def stop(self):
        PWM.stop(self.pin)
```

Abbildung 18: Programm von Lautsprecher

3.4.5 Programm für Tastenfeld

In diesem Tastenfeld nutzen wir mit der Umkehrweise. Von Anfang richten wir zuerst die Zeilen als Ausgang und die Spalten als Eingang ein. Wenn man eine Taste dauerhaft gedrückt hat, die Schaltung, die darauf gedrückte Taste hätte, wurde im Eingang Versorgung der Spannung bekommen. Nachdem man die Spalte bestimmt hat, mache man allen Pin umgekehrt. Als Schritt 2 stellen wir danach die Zeilen als Eingang und die Spalten als Ausgang ein. Mit dieser bestimmten Spalte bei wieder gedrückter Taste unterscheidet man am Ende, welcher Eingang Signal von Spannung kriegt hat. Und Jetzt kann man die Koordinate mit array Funktion die genaue Taste bestimmen.

```
class Keypad():
    def __init__(self):
        # CONSTANTS
        self.KEYPAD = [
            ["1", "2", "3"],
            ["4", "5", "6"],
            ["7", "8", "9"],
            ["*", "0", "#"]
        ]

        self.ROW         = ["P8_8", "P8_10", "P8_12", "P8_14"]
        self.COLUMN      = ["P8_16", "P8_17", "P8_18"]
        self.PASS = "1234"

    def getKey(self):
        # Set all columns as output low
        for i in range(len(self.COLUMN)):
            GPIO.setup(self.COLUMN[i], GPIO.OUT)
            GPIO.output(self.COLUMN[i], GPIO.LOW)

        # Set all rows as input
        for i in range(len(self.ROW)):
            GPIO.setup(self.ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)

        # Scan rows for pushed key/button
        # A valid key press should set "rowVal"  between 0 and 3.
        rowVal = -1
        for i in range(len(self.ROW)):
            tmpRead = GPIO.input(self.ROW[i])
            if tmpRead == 0:
                rowVal = i

        if rowVal <0 or rowVal >3:
            self.reset()
            return

        # Convert columns to input
        for j in range(len(self.COLUMN)):
            GPIO.setup(self.COLUMN[j], GPIO.IN, pull_up_down=GPIO.PUD_UP)

        # Switch the i-th row found from scan to output
        GPIO.setup(self.ROW[rowVal], GPIO.OUT)
        GPIO.output(self.ROW[rowVal], GPIO.LOW)

        # Scan columns for still-pushed key/button
        # A valid key press should set "colVal"  between 0 and 2.
        colVal = -1
        for j in range(len(self.COLUMN)):
            tmpRead = GPIO.input(self.COLUMN[j])
            if tmpRead == 0:
                colVal=j

        if colVal <0 or colVal >2:
            self.reset()
            return

        # Return the value of the key pressed
        self.reset()
        return self.KEYPAD[rowVal][colVal]

    def reset(self):
        # Reinitialize all rows and columns
        for i in range(len(self.ROW)):
            GPIO.setup(self.ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)
        for j in range(len(self.COLUMN)):
            GPIO.setup(self.COLUMN[j], GPIO.IN, pull_up_down=GPIO.PUD_UP)

    def verify_pass(self, pass_entered):
        return pass_entered == self.PASS
```

Abbildung 19: Programm von Tastenfeld

Zusammenfassung

In diesem Projekt konnte man sich sehr viele Kenntnisse, Fähigkeiten und Fertigkeiten über das Linux-Betriebssystem, Bussysteme und die Python Programmiersprache aneignen. Obwohl nur einige Sensoren realisiert wurde, begegnete man allerdings vielen Problemen, z. B. wie schreibt man Programm für jeden Sensor mit class- Form, wie nutzt man Polling Funktion, hier in unseres Programm callback ist, wie man in einzelnem Prozess die Reihenfolge von Befehlen anpassen soll, wie muss ein geeignetes Überwachungsprogramm entwickelt werden usw. Die endliche Lösung wurden viel Aufmerksamkeit und Zeit dafür aufgewendet, eine richtige Lösung zu finden. Man musste erkennen, dass bei den praktischen Anwendungen von Geräten immer unerwartete Probleme auftreten können. Geduld und Ernst sind nötig, wenn man versucht, Probleme zu lösen. Das Praktikum bietet auch eine gute Chance, das theoretische Wissen aus dem Studium in die Praxis umzusetzen. Den beiden Mitarbeitern Müller, Stange und auch Professor Chmielewski gebührt ein großes Dankeschön.

Abbildungsverzeichnis