Arduino Secret Pattern Lock

Generated by Doxygen 1.8.11

Contents

1 Hierarchical Index **2**

1.1 Class Hierarchy 2

2 Class Index **2**

2.1 Class List 2

3 File Index **3**

3.1 File List 3

4 Class Documentation **3**

4.1 Feedbacker Class Reference 3

4.1.1 Detailed Description 4

4.1.2 Member Enumeration Documentation 4

4.1.3 Member Function Documentation 4

4.2 KnockLock Class Reference 5

4.2.1 Detailed Description 6

4.2.2 Member Enumeration Documentation 6

4.2.3 Constructor & Destructor Documentation 6

4.2.4 Member Function Documentation 6

4.2.5 Member Data Documentation 8

4.3 LedFeedbacker Class Reference 9

4.3.1 Detailed Description 9

4.3.2 Constructor & Destructor Documentation 10

4.3.3 Member Function Documentation 10

4.3.4 Member Data Documentation 10

5 File Documentation **10**

5.1 Feedbacker.cpp File Reference 10

5.2 Feedbacker.cpp 11

5.3 Feedbacker.h File Reference 11

5.4 Feedbacker.h 11

5.5 KnockLock.cpp File Reference 12

5.6 KnockLock.cpp 12

5.7 KnockLock.h File Reference 14

5.7.1 Macro Definition Documentation 15

5.8 KnockLock.h 16

5.9 LedFeedbacker.cpp File Reference 17

5.10 LedFeedbacker.cpp 17

5.11 LedFeedbacker.h File Reference 18

5.12 LedFeedbacker.h 19

Index **21**

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Feedbacker **3**

LedFeedbacker **9**

KnockLock **5**

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Feedbacker
Arduino - ArduinoSecretPatternLock driver **3**

KnockLock **5**

LedFeedbacker 9

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

Feedbacker.cpp 10

Feedbacker.h 11

KnockLock.cpp 12

KnockLock.h 14

LedFeedbacker.cpp 17

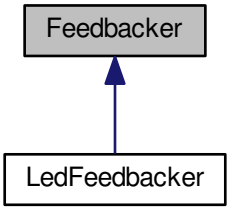
LedFeedbacker.h 18

4 Class Documentation

4.1 Feedbacker Class Reference

`#include <Feedbacker.h>`

Inheritance diagram for Feedbacker:



Public Types

- enum Feedback {
  ACCESS_DENIED = 0x00, ACCESS_GRANTED, INVALID_OPERATION, ENTER_PROGRAM_MODE,
  EXIT_PROGRAM_MODE, SUCCESSFULLY_SAVED }

Public Member Functions

- virtual void sendFeedback (Feedback feedback)=0

4.1.1 Detailed Description

Arduino - ArduinoSecretPatternLock driver.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 10 of file Feedbacker.h.

4.1.2 Member Enumeration Documentation

4.1.2.1 enum Feedbacker::Feedback

Enumerator

ACCESS_DENIED

ACCESS_GRANTED

INVALID_OPERATION

ENTER_PROGRAM_MODE

EXIT_PROGRAM_MODE

SUCCESSFULLY_SAVED

Definition at line 13 of file Feedbacker.h.

4.1.3 Member Function Documentation

4.1.3.1 virtual void Feedbacker::sendFeedback (Feedback *feedback*) [pure virtual]

Implemented in LedFeedbacker.

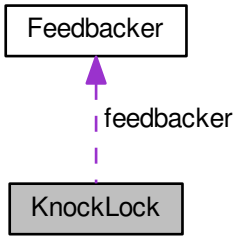
The documentation for this class was generated from the following file:

- Feedbacker.h

4.2 KnockLock Class Reference

`#include <KnockLock.h>`

Collaboration diagram for KnockLock:



Public Member Functions

- KnockLock (int knockPin, int programPin, int buzzerPin, Feedbacker *feedbacker, RandomAccess *storage)
- void initialize ()
- void playback ()
- bool listen (unsigned long timeout=0)

Private Types

- enum Feedback { ENTER_PROGRAM_MODE = 0x00, EXIT_PROGRAM_MODE, PATTERN_SAVED, KNOCKS_MISMATCH }

Private Member Functions

- void enterProgramMode ()
- void exitProgramMode ()
- bool isInProgramMode ()
- void processKnocks ()
- void knockDelay ()
- void normalizeKnocks ()
- bool validateKnocks ()
- bool doKnocksMatchPattern ()
- void loadPattern ()
- void savePattern ()

Private Attributes

- int knockPin
- int programPin
- int buzzerPin
- Feedbacker * feedbacker
- RandomAccess * storage
- int patternSize
- int knocksSize
- bool programMode
- unsigned char pattern [KNOCK_LOCK_MAX_KNOCKS]
- unsigned int knocks [KNOCK_LOCK_MAX_KNOCKS]

4.2.1 Detailed Description

Definition at line 16 of file KnockLock.h.

4.2.2 Member Enumeration Documentation

4.2.2.1 enum KnockLock::Feedback [private]

Enumerator

ENTER_PROGRAM_MODE

EXIT_PROGRAM_MODE

PATTERN_SAVED

KNOCKS_MISMATCH

Definition at line 38 of file KnockLock.h.

4.2.3 Constructor & Destructor Documentation

4.2.3.1 KnockLock::KnockLock (int *knockPin*, int *programPin*, int *buzzerPin*, Feedbacker * *feedbacker*, RandomAccess * *storage*)

Definition at line 4 of file KnockLock.cpp.

4.2.4 Member Function Documentation

4.2.4.1 bool KnockLock::doKnocksMatchPattern ( ) [private]

Definition at line 98 of file KnockLock.cpp.

4.2.4.2 void KnockLock::enterProgramMode ( ) [private]

Enter in the program mode.

Keep programPin in high impedance for KNOCK_LOCK_PROGRAM_BUTTON_MIN_TIME_PRESSED millis.

Definition at line 115 of file KnockLock.cpp.

4.2.4.3 void KnockLock::exitProgramMode ( ) [private]

Exits from program mode.

After record the patter.

Definition at line 129 of file KnockLock.cpp.

4.2.4.4 void KnockLock::initialize ( )

Definition at line 8 of file KnockLock.cpp.

4.2.4.5 bool KnockLock::isInProgramMode ( ) [private]

Definition at line 134 of file KnockLock.cpp.

4.2.4.6 void KnockLock::knockDelay ( ) [private]

Definition at line 138 of file KnockLock.cpp.

4.2.4.7 bool KnockLock::listen (unsigned long *timeout* = 0)

Definition at line 31 of file KnockLock.cpp.

4.2.4.8 void KnockLock::loadPattern ( ) [private]

Definition at line 147 of file KnockLock.cpp.

4.2.4.9 void KnockLock::normalizeKnocks ( ) [private]

Definition at line 71 of file KnockLock.cpp.

4.2.4.10 void KnockLock::playback ( )

Definition at line 15 of file KnockLock.cpp.

4.2.4.11 void KnockLock::processKnocks ( ) [private]

After the first knock it records the times between knocks.

Definition at line 53 of file KnockLock.cpp.

4.2.4.12 void KnockLock::savePattern ( ) [private]

Definition at line 161 of file KnockLock.cpp.

4.2.4.13 bool KnockLock::validateKnocks ( ) [private]

Definition at line 83 of file KnockLock.cpp.

4.2.5 Member Data Documentation

4.2.5.1 int KnockLock::buzzerPin [private]

Definition at line 22 of file KnockLock.h.

4.2.5.2 Feedbacker* KnockLock::feedbacker [private]

Definition at line 24 of file KnockLock.h.

4.2.5.3 int KnockLock::knockPin [private]

Definition at line 18 of file KnockLock.h.

4.2.5.4 unsigned int KnockLock::knocks[KNOCK_LOCK_MAX_KNOCKS] [private]

Definition at line 36 of file KnockLock.h.

4.2.5.5 int KnockLock::knocksSize [private]

Definition at line 30 of file KnockLock.h.

4.2.5.6 unsigned char KnockLock::pattern[KNOCK_LOCK_MAX_KNOCKS] [private]

Definition at line 34 of file KnockLock.h.

4.2.5.7 int KnockLock::patternSize [private]

Definition at line 28 of file KnockLock.h.

4.2.5.8 bool KnockLock::programMode [private]

Definition at line 32 of file KnockLock.h.

4.2.5.9 int KnockLock::programPin [private]

Definition at line 20 of file KnockLock.h.

4.2.5.10 RandomAccess* KnockLock::storage [private]

Definition at line 26 of file KnockLock.h.

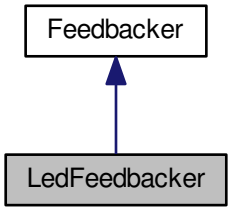
The documentation for this class was generated from the following files:

- KnockLock.h
- KnockLock.cpp

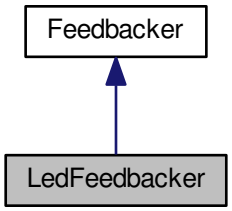
4.3 LedFeedbacker Class Reference

`#include <LedFeedbacker.h>`

Inheritance diagram for LedFeedbacker:



Collaboration diagram for LedFeedbacker:



Public Member Functions

- LedFeedbacker (int ledPin)
- void sendFeedback (Feedback feedback)

Private Attributes

- int ledPin

Additional Inherited Members

4.3.1 Detailed Description

Definition at line 3 of file LedFeedbacker.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 LedFeedbacker::LedFeedbacker (int *ledPin*)

Definition at line 4 of file LedFeedbacker.cpp.

4.3.3 Member Function Documentation

4.3.3.1 void LedFeedbacker::sendFeedback (Feedback *feedback*) [virtual]

Implements Feedbacker.

Definition at line 8 of file LedFeedbacker.cpp.

4.3.4 Member Data Documentation

4.3.4.1 int LedFeedbacker::ledPin [private]

Definition at line 5 of file LedFeedbacker.h.

The documentation for this class was generated from the following files:

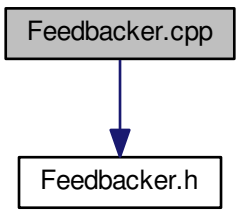
- LedFeedbacker.h
- LedFeedbacker.cpp

5 File Documentation

5.1 Feedbacker.cpp File Reference

```
#include "Feedbacker.h"
```

Include dependency graph for Feedbacker.cpp:

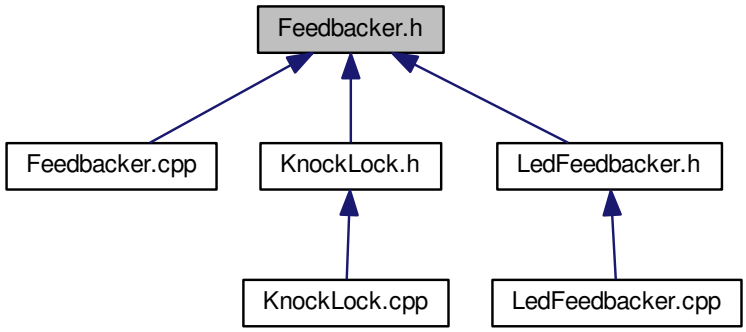


5.2 Feedbacker.cpp

```
00001 #include "Feedbacker.h"
```

5.3 Feedbacker.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Feedbacker

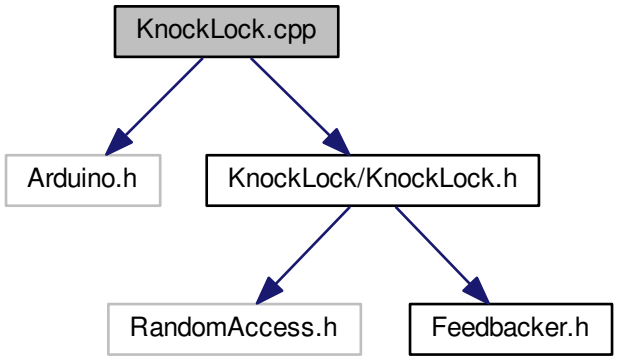
5.4 Feedbacker.h

```
00001
00007 #ifndef __ARDUINO_SECRET_PATTERN_LOCK_FEEDBACKER_H__
00008 #define __ARDUINO_SECRET_PATTERN_LOCK_FEEDBACKER_H__ 1
00009
00010 class Feedbacker {
00011 public:
00012
00013     enum Feedback {
00014         ACCESS_DENIED = 0x00,
00015         ACCESS_GRANTED,
00016         INVALID_OPERATION,
00017         ENTER_PROGRAM_MODE,
00018         EXIT_PROGRAM_MODE,
00019         SUCCESSFULLY_SAVED
00020     };
00021
00022     virtual void sendFeedback(Feedback feedback) = 0;
00023 };
00024
00025 #endif // __ARDUINO_SECRET_PATTERN_LOCK_FEEDBACKER_H__
```

5.5 KnockLock.cpp File Reference

```
#include <Arduino.h>
#include <KnockLock/KnockLock.h>
```

Include dependency graph for KnockLock.cpp:



5.6 KnockLock.cpp

```
00001 #include <Arduino.h>
00002 #include <KnockLock/KnockLock.h>
00003
00004 KnockLock::KnockLock(int knockPin, int programPin, int buzzerPin,
      Feedbacker *feedbacker, RandomAccess *storage)
00005         : knockPin(knockPin), programPin(programPin), buzzerPin(buzzerPin), feedbacker(feedbacker), storage
      (storage), patternSize(0), knocksSize(0), programMode(false) {
00006 }
00007
00008 void KnockLock::initialize() {
00009     pinMode(knockPin, INPUT);
00010     pinMode(programPin, INPUT);
00011     pinMode(buzzerPin, OUTPUT);
00012     loadPattern();
00013 }
00014
00015 void KnockLock::playback() {
00016     for (unsigned char i = 0; i < patternSize; i++) {
00017         for (unsigned int j = 0; j < 10; j++) {
00018             digitalWrite(buzzerPin, HIGH);
00019             delay(10);
00020             digitalWrite(buzzerPin, LOW);
00021         }
00022         delay(pattern[i] * 10);
00023     }
00024     for (unsigned int j = 0; j < 10; j++) {
00025         digitalWrite(buzzerPin, HIGH);
00026         delay(10);
00027         digitalWrite(buzzerPin, LOW);
00028     }
00029 }
00030
00031 bool KnockLock::listen(unsigned long timeout) {
00032     unsigned long startTime = millis();
00033     do {
00034         if (digitalRead(programPin) == HIGH) {
00035             enterProgramMode();
00036         }
00037         if (digitalRead(knockPin) == HIGH) {
00038             processKnocks();
00039             if (validateKnocks()) {
00040                 feedbacker->sendFeedback(
```

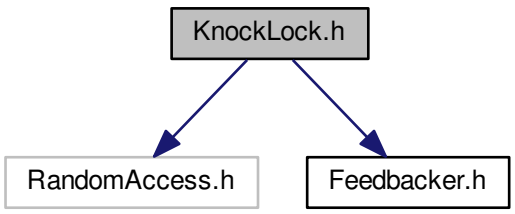
```
      Feedbacker::ACCESS_GRANTED);
00041                 return true;
00042             }
00043             if (isInProgramMode()) {
00044                 exitProgramMode();
00045             } else {
00046                 feedbacker->sendFeedback(
      Feedbacker::ACCESS_DENIED);
00047             }
00048         }
00049     } while (timeout == 0 || millis() - startTime < timeout);
00050     return false;
00051 }
00052
00053 void KnockLock::processKnocks() {
00054     unsigned long startTime, now;
00055     knocksSize = 0;
00056     now = millis();
00057     startTime = now;
00058     knockDelay();
00059     do {
00060         unsigned char knock = digitalRead(knockPin);
00061         if (knock == HIGH) {
00062             knocks[knocksSize] = now - startTime;
00063             knocksSize++;
00064             startTime = now;
00065             knockDelay();
00066         }
00067         now = millis();
00068     } while ((now - startTime < KNOCK_LOCK_MAX_TIME_BETWEEN_KNOCKS) && (
      knocksSize < KNOCK_LOCK_MAX_KNOCKS));
00069 }
00070
00071 void KnockLock::normalizeKnocks() {
00072     unsigned int maxPatternTime = 0;
00073     for (int i = 0; i < knocksSize; i++) {
00074         if (knocks[i] > maxPatternTime) {
00075             maxPatternTime = knocks[i];
00076         }
00077     }
00078     for (int i = 0; i < knocksSize; i++) {
00079         knocks[i] = map(knocks[i], 0, maxPatternTime, 0,
      KNOCK_LOCK_NORMALIZATION_LENGTH);
00080     }
00081 }
00082
00083 bool KnockLock::validateKnocks() {
00084     normalizeKnocks();
00085     if (knocksSize == 0) {
00086         return false;
00087     }
00088     if (isInProgramMode()) {
00089         savePattern();
00090         return false;
00091     }
00092     if (knocksSize != patternSize) {
00093         return false;
00094     }
00095     return doKnocksMatchPattern();
00096 }
00097
00098 bool KnockLock::doKnocksMatchPattern() {
00099     int totalTimeMissmatch = 0;
00100     int timeMissmatch = 0;
00101     for (int i = 0; i < patternSize; i++) {
00102         timeMissmatch = abs(((int)knocks[i]) - pattern[i]);
00103         Serial.println(timeMissmatch);
00104         if (timeMissmatch > KNOCK_LOCK_TIME_MISSMATCH_THRESHOLD) {
00105             return false;
00106         }
00107         totalTimeMissmatch += timeMissmatch;
00108     }
00109     if (totalTimeMissmatch / patternSize >
      KNOCK_LOCK_AVG_TIME_MISSMATCH_THRESHOLD) {
00110         return false;
00111     }
00112     return true;
00113 }
00114
00115 void KnockLock::enterProgramMode() {
00116     unsigned long startTime = millis();
00117     do {
00118         if (digitalRead(programPin) == LOW) {
00119             return;
00120         }
00121     } while ((millis() - startTime) < KNOCK_LOCK_PROGRAM_BUTTON_MIN_TIME_PRESSED
      );
```

```
00122     programMode = true;
00123     feedbacker->sendFeedback(
      Feedbacker::ENTER_PROGRAM_MODE);
00124     while (digitalRead(programPin) == HIGH) {
00125     }
00126     delay(KNOCK_LOCK_RELEASE_BUTTON_DEBOUNCE_TIME);
00127 }
00128
00129 void KnockLock::exitProgramMode() {
00130     programMode = false;
00131     feedbacker->sendFeedback(Feedbacker::EXIT_PROGRAM_MODE
      );
00132 }
00133
00134 bool KnockLock::isInProgramMode() {
00135     return programMode;
00136 }
00137
00138 void KnockLock::knockDelay() {
00139     int itterations = KNOCK_LOCK_DISSIPATE_KNOCK_DEBOUNCE_TIME / 20
      ;
00140     for (int i = 0; i < itterations; i++) {
00141         delay(10);
00142         analogRead(knockPin);
00143         delay(10);
00144     }
00145 }
00146
00147 void KnockLock::loadPattern() {
00148     storage->seek(0);
00149     patternSize = storage->readUnsignedChar();
00150     if (patternSize < 0) {
00151         patternSize = 0;
00152     }
00153     if (patternSize > KNOCK_LOCK_MAX_KNOCKS) {
00154         patternSize = KNOCK_LOCK_MAX_KNOCKS;
00155     }
00156     for (int i = 0; i < patternSize; i++) {
00157         pattern[i] = storage->readUnsignedChar();
00158     }
00159 }
00160
00161 void KnockLock::savePattern() {
00162     storage->seek(0);
00163     storage->writeUnsignedChar(knocksSize);
00164     for (int i = 0; i < knocksSize; i++) {
00165         storage->writeUnsignedChar(knocks[i]);
00166         pattern[i] = knocks[i];
00167     }
00168     patternSize = knocksSize;
00169     feedbacker->sendFeedback(
      Feedbacker::SUCCESSFULLY_SAVED);
00170     playback();
00171 }
```

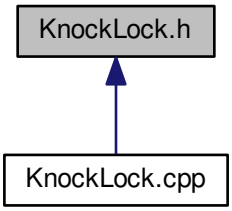
5.7 KnockLock.h File Reference

```
#include <RandomAccess.h>
#include <Feedbacker.h>
```

Include dependency graph for KnockLock.h:



This graph shows which files directly or indirectly include this file:



Classes

- class KnockLock

Macros

- #define KNOCK_LOCK_MAX_KNOCKS 16
- #define KNOCK_LOCK_MAX_TIME_BETWEEN_KNOCKS 1000
- #define KNOCK_LOCK_DISSIPATE_KNOCK_DEBOUNCE_TIME 100
- #define KNOCK_LOCK_RELEASE_BUTTON_DEBOUNCE_TIME 500
- #define KNOCK_LOCK_PROGRAM_BUTTON_MIN_TIME_PRESSED 1500
- #define KNOCK_LOCK_NORMALIZATION_LENGTH 100
- #define KNOCK_LOCK_PATTERN_SAVED_BLINKS 3
- #define KNOCK_LOCK_TIME_MISSMATCH_THRESHOLD 25
- #define KNOCK_LOCK_AVG_TIME_MISSMATCH_THRESHOLD 15

5.7.1 Macro Definition Documentation

5.7.1.1 #define KNOCK_LOCK_AVG_TIME_MISSMATCH_THRESHOLD 15

Definition at line 14 of file KnockLock.h.

5.7.1.2 #define KNOCK_LOCK_DISSIPATE_KNOCK_DEBOUNCE_TIME 100

Definition at line 6 of file KnockLock.h.

5.7.1.3 #define KNOCK_LOCK_MAX_KNOCKS 16

Definition at line 4 of file KnockLock.h.

5.7.1.4 #define KNOCK_LOCK_MAX_TIME_BETWEEN_KNOCKS 1000

Definition at line 5 of file KnockLock.h.

5.7.1.5 #define KNOCK_LOCK_NORMALIZATION_LENGTH 100

Definition at line 11 of file KnockLock.h.

5.7.1.6 #define KNOCK_LOCK_PATTERN_SAVED_BLINKS 3

Definition at line 12 of file KnockLock.h.

5.7.1.7 #define KNOCK_LOCK_PROGRAM_BUTTON_MIN_TIME_PRESSED 1500

Definition at line 8 of file KnockLock.h.

5.7.1.8 #define KNOCK_LOCK_RELEASE_BUTTON_DEBOUNCE_TIME 500

Definition at line 7 of file KnockLock.h.

5.7.1.9 #define KNOCK_LOCK_TIME_MISSMATCH_THRESHOLD 25

Definition at line 13 of file KnockLock.h.

5.8 KnockLock.h

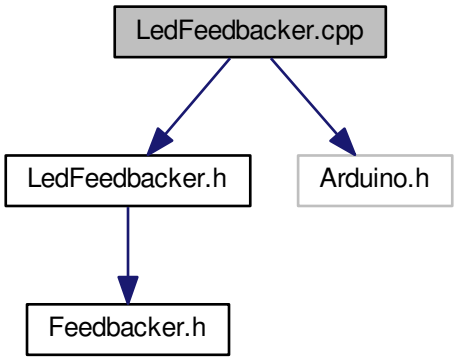
```
00001 #include <RandomAccess.h>
00002 #include <Feedbacker.h>
00003
00004 #define KNOCK_LOCK_MAX_KNOCKS                           16
00005 #define KNOCK_LOCK_MAX_TIME_BETWEEN_KNOCKS              1000
00006 #define KNOCK_LOCK_DISSIPATE_KNOCK_DEBOUNCE_TIME        100
00007 #define KNOCK_LOCK_RELEASE_BUTTON_DEBOUNCE_TIME         500
00008 #define KNOCK_LOCK_PROGRAM_BUTTON_MIN_TIME_PRESSED      1500
00009
00010 // Shuldn't be bigger than sizeof(char)
00011 #define KNOCK_LOCK_NORMALIZATION_LENGTH                 100
00012 #define KNOCK_LOCK_PATTERN_SAVED_BLINKS                 3
00013 #define KNOCK_LOCK_TIME_MISSMATCH_THRESHOLD             25
00014 #define KNOCK_LOCK_AVG_TIME_MISSMATCH_THRESHOLD         15
00015
00016 class KnockLock {
00017
00018     int knockPin;
00019
00020     int programPin;
00021
00022     int buzzerPin;
00023
00024     Feedbacker *feedbacker;
00025
00026     RandomAccess *storage;
00027
00028     int patternSize;
00029
00030     int knocksSize;
00031
00032     bool programMode;
00033
00034     unsigned char pattern[KNOCK_LOCK_MAX_KNOCKS];
00035
00036     unsigned int knocks[KNOCK_LOCK_MAX_KNOCKS];
00037
00038     enum Feedback {
00039         ENTER_PROGRAM_MODE = 0x00,
00040         EXIT_PROGRAM_MODE,
00041         PATTERN_SAVED,
00042         KNOCKS_MISMATCH
00043     };
00044 public:
00045
00046     KnockLock(int knockPin, int programPin, int buzzerPin, Feedbacker *feedbacker,
      RandomAccess *storage);
```

```
00047
00048     void initialize();
00049
00050     void playback();
00051
00052     bool listen(unsigned long timeout = 0);
00053
00054 private:
00055
00060     void enterProgramMode();
00061
00065     void exitProgramMode();
00066
00067     bool isInProgramMode();
00068
00072     void processKnocks();
00073
00074     void knockDelay();
00075
00076     void normalizeKnocks();
00077
00078     bool validateKnocks();
00079
00080     bool doKnocksMatchPattern();
00081
00082     void loadPattern();
00083
00084     void savePattern();
00085 };
```

5.9 LedFeedbacker.cpp File Reference

```
#include "LedFeedbacker.h"
#include <Arduino.h>
```

Include dependency graph for LedFeedbacker.cpp:



5.10 LedFeedbacker.cpp

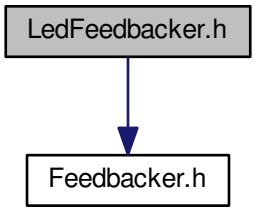
```
00001 #include "LedFeedbacker.h"
00002 #include <Arduino.h>
00003
00004 LedFeedbacker::LedFeedbacker(int ledPin) : ledPin(ledPin) {
00005     pinMode(ledPin, OUTPUT);
00006 }
00007
00008 void LedFeedbacker::sendFeedback(Feedback feedback) {
00009     unsigned char blinks = 0;
```

```
00010     switch (feedback) {
00011     case ENTER_PROGRAM_MODE:
00012         digitalWrite(ledPin, HIGH);
00013         break;
00014     case EXIT_PROGRAM_MODE:
00015         digitalWrite(ledPin, LOW);
00016         break;
00017     case ACCESS_GRANTED:
00018         blinks = 1;
00019         break;
00020     case ACCESS_DENIED:
00021         blinks = 2;
00022         break;
00023     case SUCCESSFULLY_SAVED:
00024         blinks = 4;
00025         break;
00026     case INVALID_OPERATION:
00027         break;
00028     }
00029     for (unsigned char i = 0; i < blinks; i++) {
00030         digitalWrite(ledPin, HIGH);
00031         delay(100);
00032         digitalWrite(ledPin, LOW);
00033         delay(100);
00034     }
00035 }
```

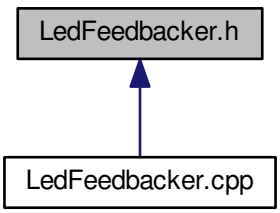
5.11 LedFeedbacker.h File Reference

`#include <Feedbacker.h>`

Include dependency graph for LedFeedbacker.h:



This graph shows which files directly or indirectly include this file:



Classes

- class LedFeedbacker

5.12 LedFeedbacker.h

```
00001 #include <Feedbacker.h>
00002
00003 class LedFeedbacker : public Feedbacker {
00004
00005     int ledPin;
00006 public:
00007
00008     LedFeedbacker(int ledPin);
00009
00010     void sendFeedback(Feedback feedback);
00011 };
```

Index

ACCESS_DENIED
  Feedbacker, 4
ACCESS_GRANTED
  Feedbacker, 4

buzzerPin
  KnockLock, 8

doKnocksMatchPattern
  KnockLock, 6

ENTER_PROGRAM_MODE
  Feedbacker, 4
  KnockLock, 6
EXIT_PROGRAM_MODE
  Feedbacker, 4
  KnockLock, 6
enterProgramMode
  KnockLock, 6
exitProgramMode
  KnockLock, 6

Feedback
  Feedbacker, 4
  KnockLock, 6
Feedbacker, 3
  ACCESS_DENIED, 4
  ACCESS_GRANTED, 4
  ENTER_PROGRAM_MODE, 4
  EXIT_PROGRAM_MODE, 4
  Feedback, 4
  INVALID_OPERATION, 4
  SUCCESSFULLY_SAVED, 4
  sendFeedback, 4
feedbacker
  KnockLock, 8
Feedbacker.cpp, 10, 11
Feedbacker.h, 11

INVALID_OPERATION
  Feedbacker, 4
initialize
  KnockLock, 7
isInProgramMode
  KnockLock, 7

KNOCK_LOCK_AVG_TIME_MISSMATCH_THRESH↩
    OLD
  KnockLock.h, 15
KNOCK_LOCK_DISSIPATE_KNOCK_DEBOUNCE_↩
    TIME
  KnockLock.h, 15
KNOCK_LOCK_MAX_KNOCKS
  KnockLock.h, 15
KNOCK_LOCK_MAX_TIME_BETWEEN_KNOCKS
  KnockLock.h, 15
KNOCK_LOCK_NORMALIZATION_LENGTH
  KnockLock.h, 15
KNOCK_LOCK_PATTERN_SAVED_BLINKS
  KnockLock.h, 16
KNOCK_LOCK_PROGRAM_BUTTON_MIN_TIME_↩
    PRESSED
  KnockLock.h, 16
KNOCK_LOCK_RELEASE_BUTTON_DEBOUNCE_↩
    TIME
  KnockLock.h, 16
KNOCK_LOCK_TIME_MISSMATCH_THRESHOLD
  KnockLock.h, 16
KNOCKS_MISMATCH
  KnockLock, 6
knockDelay
  KnockLock, 7
KnockLock, 5
  buzzerPin, 8
  doKnocksMatchPattern, 6
  ENTER_PROGRAM_MODE, 6
  EXIT_PROGRAM_MODE, 6
  enterProgramMode, 6
  exitProgramMode, 6
  Feedback, 6
  feedbacker, 8
  initialize, 7
  isInProgramMode, 7
  KNOCKS_MISMATCH, 6
  knockDelay, 7
  KnockLock, 6
  knockPin, 8
  knocks, 8
  knocksSize, 8
  listen, 7
  loadPattern, 7
  normalizeKnocks, 7
  PATTERN_SAVED, 6
  pattern, 8
  patternSize, 8
  playback, 7
  processKnocks, 7
  programMode, 8
  programPin, 8
  savePattern, 7
  storage, 8
  validateKnocks, 7
KnockLock.cpp, 12
KnockLock.h, 14, 16
  KNOCK_LOCK_AVG_TIME_MISSMATCH_TH↩
    RESHOLD, 15
  KNOCK_LOCK_DISSIPATE_KNOCK_DEBOU↩
    NCE_TIME, 15
  KNOCK_LOCK_MAX_KNOCKS, 15
  KNOCK_LOCK_MAX_TIME_BETWEEN_KNOC↩
    KS, 15
  KNOCK_LOCK_NORMALIZATION_LENGTH, 15

KNOCK_LOCK_PATTERN_SAVED_BLINKS, 16
KNOCK_LOCK_PROGRAM_BUTTON_MIN_TI↩
ME_PRESSED, 16
KNOCK_LOCK_RELEASE_BUTTON_DEBOUN↩
CE_TIME, 16
KNOCK_LOCK_TIME_MISSMATCH_THRESH↩
OLD, 16
knockPin
KnockLock, 8
knocks
KnockLock, 8
knocksSize
KnockLock, 8

LedFeedbacker, 9
LedFeedbacker, 10
ledPin, 10
sendFeedback, 10
LedFeedbacker.cpp, 17
LedFeedbacker.h, 18, 19
ledPin
LedFeedbacker, 10
listen
KnockLock, 7
loadPattern
KnockLock, 7

normalizeKnocks
KnockLock, 7

PATTERN_SAVED
KnockLock, 6
pattern
KnockLock, 8
patternSize
KnockLock, 8
playback
KnockLock, 7
processKnocks
KnockLock, 7
programMode
KnockLock, 8
programPin
KnockLock, 8

SUCCESSFULLY_SAVED
Feedbacker, 4
savePattern
KnockLock, 7
sendFeedback
Feedbacker, 4
LedFeedbacker, 10
storage
KnockLock, 8

validateKnocks
KnockLock, 7