Search the Arduino Playground

Manuals and Curriculum
(http://playground.arduino.cc/Main/ManualsAndCurriculum)
Arduino StackExchange
(http://arduino.stackexchange.com)
Board Setup and Configuration
(http://playground.arduino.cc/Main/ArduinoCoreHardware)
Development Tools
(http://playground.arduino.cc/Main/DevelopmentTools)
Arduino on other Chips
(http://playground.arduino.cc/Main/ArduinoOnOtherAtmelChips)
Interfacing With Hardware
(http://playground.arduino.cc/Main/InterfacingWithHardware)

- Output (http://playground.arduino.cc/Main/InterfacingWithHardware#Output)
- Input (http://playground.arduino.cc/Main/InterfacingWithHardware#InputTOC)
- User Interface (http://playground.arduino.cc/Main/InterfacingWithHardware#ui)
- Storage (http://playground.arduino.cc/Main/InterfacingWithHardware#Storage)
- Communication (http://playground.arduino.cc/Main/InterfacingWithHardware#Communication)
- Power supplies (http://playground.arduino.cc/Main/IntWithHW-PwrSup)
- General (http://playground.arduino.cc/Main/InterfacingWithHardware#General)

Interfacing with Software
(http://playground.arduino.cc/Main/InterfacingWithSoftware)
User Code Library
(http://playground.arduino.cc/Main/GeneralCodeLibrary)

- Snippets and Sketches (http://playground.arduino.cc/Main/SketchList)
- Libraries (http://playground.arduino.cc/Main/LibraryList)
- Tutorials (http://playground.arduino.cc/Main/TutorialList)

Suggestions & Bugs
(https://github.com/arduino/arduino/issues)
Electronics Technique
(http://playground.arduino.cc/Main/ElectroInfoResources)
Sources for Electronic Parts
(http://playground.arduino.cc/Main/Resources)

Related Hardware and Initiatives
(http://playground.arduino.cc/Main/SimilarBoards)
Arduino People/Groups & Sites
(http://playground.arduino.cc/Main/People)
Exhibition
(http://playground.arduino.cc/Projects/ArduinoUsers)
Project Ideas
(http://playground.arduino.cc/Projects/Ideas)
Languages
(http://playground.arduino.cc/Main/Languages)

Participate
(http://playground.arduino.cc/Main/Participate)

- Formatting guidelines
  (http://playground.arduino.cc/Main/Participate#contribrules)
- All recent changes
  (http://playground.arduino.cc/Site/AllRecentChanges)
- PmWiki
  (http://playground.arduino.cc/PmWiki/PmWiki)
- WikiSandBox training
  (http://playground.arduino.cc/Main/WikiSandbox)
- Basic Editing
  (http://playground.arduino.cc/PmWiki/BasicEditing)
- Documentation index
  (http://www.pmwiki.org/wiki/PmWiki/DocumentationIndex)

```
Keypad Library for Arduino

Authors:  Mark Stanley, Alexander Brevig
Contact: mstanley@technologist.com
Contact: alexanderbrevig@gmail.com
```

```
I am Alexander, the lead developer for libraries
that ship with the Wiring distribution. I will be
maintaining my libraries here:
http://wiring.uniandes.edu.co/source/trunk/wiring/

As of version 1.0 - Wiring supports Arduino boards.
You are welcome to check it out! Please visit
http://wiring.org.co/download/
```

Navigation

Current version

3.0 2012-07-12 - Mark Stanley : Made library multi-keypress by default. (Backwards compatible)
2015-09-18 - Code has been replicated to GitHub for IDE's 1.6.2 and above. Small changes have been made to conform with the latest library specification, but the functional code is intact. To notify of changes or raise issues visit: https://github.com/Chris--A/Keypad (https://github.com/Chris--A/Keypad)

History

3.0 2015-09-18 - Christopher Andrews : Copied to GitHub (https://github.com/Chris--A/Keypad) for modern IDE library manager.
3.0 2012-07-12 - Mark Stanley : Modified pin functions to support Keypad_I2C
3.0 2012-07-12 - Stanley & Young : Fix for multiple keypad objects.
3.0 2012-07-12 - Mark Stanley : Fixed bug that caused shorted pins.
2.0 2011-12-29 - Mark Stanley : Added Nick Gammon's changes. (http://arduino.cc/forum/index.php?topic=58337.0)
2.0 2011-12-29 - Mark Stanley : Added waitForKey()
2.0 2011-12-23 - Mark Stanley : Rewrote state machine.
2.0 2011-12-23 - Mark Stanley : Significant speed improvements.
1.8 2011-11-29 - Tom Putzeys : Enabled internal pull-ups on non-active columns
1.8 2011-11-21 - Mark Stanley : Added test for version 1.0 of the IDE
1.8 2009-07-08 - Alexander Brevig : Added no restrictions on sizes or keymaps
1.8 2009-07-08 - Alexander Brevig : Added no restrictions on sizes or keymaps
See source files for a complete change history.

Description

Keypad is a library for using *matrix* style keypads with the Arduino. As of version 3.0 it now supports mulitple keypresses.

This library is based upon the Keypad Tutorial (http://playground.arduino.cc/Main/KeypadTutorial).

It was created to promote Hardware Abstraction. It improves readability of the code by hiding the pinMode and digitalRead calls for the user.

Keypad library is part of the Hardware Abstraction (http://playground.arduino.cc/Code/HardwareAbstraction) libraries.

Version 3.0 has just been posted (19 July 2012) and was rewritten to support multi-keypresses by default. But for those who still need the original single-keypress functionality, the library is fully backwards compatible.

You won't need external resistors or diodes because the library uses the internal pullup resistors and additonally ensures that all unused column pins are high-impedance.

Support was added to allow other hardware to be used along with a keypad. *Joe Young's keypad library added support for several I2C expander chips.* You can find it here: https://github.com/joeyoung/arduino_keypads (https://github.com/joeyoung/arduino_keypads)

Download, install and import

This library is now available via the Arduino IDE library manager. If you are using a modern IDE (1.6.2 or above), you can simply use the menu:

Sketch->**Include Library**->**Manage Libraries**... Then search for *Keypad*.

Once found, click on its entry and the install button will appear. The zip file below is for the retro IDE's (not recommended for use, upgrade!).

Download here: keypad.zip (http://playground.arduino.cc/uploads/Code/keypad.zip)

Put the Keypad folder in "*arduino*\libraries\".
In the Arduino IDE, create a new sketch (or open one) and select from the menubar "Sketch -> Import Library -> Keypad".
Once the library is imported, an "#include <Keypad.h>" line will appear at the top of your Sketch.

Creation

Constructors:

1. Keypad(makeKeymap(userKeymap), row[], col[], rows, cols)

```
const byte rows = 4; //four rows
const byte cols = 3; //three columns
char keys[rows][cols] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'#','0','*'}
};
byte rowPins[rows] = {5, 4, 3, 2}; //connect to the row pinouts of the keypad
byte colPins[cols] = {8, 7, 6}; //connect to the column pinouts of the keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols );
```

Instantiates a Keypad object that uses pins 5, 4, 3, 2 as row pins, and 8, 7, 6 as column pins.
This keypad has 4 rows and 3 columns, resulting in 12 keys.

Functions

void begin(makeKeymap(userKeymap))

Initializes the internal keymap to be equal to userKeymap
[See File -> Examples -> Keypad -> Examples -> CustomKeypad]

char waitForKey()

This function will wait forever until someone presses a key. **Warning:** It blocks all other code until a key is pressed. That means no blinking LED's, no LCD screen updates, no nothing with the exception of interrupt routines.

char getKey()

Returns the key that is pressed, if any. This function is non-blocking.

KeyState getState()

Returns the current state of any of the keys.
The four states are IDLE, PRESSED, RELEASED and HOLD.

boolean keyStateChanged()

New in version 2.0: Let's you know when the key has changed from one state to another. For example, instead of just testing for a valid key you can test for when a key was pressed.

setHoldTime(unsigned int time)

Set the amount of milliseconds the user will have to hold a button until the HOLD state is triggered.

setDebounceTime(unsigned int time)

Set the amount of milliseconds the keypad will wait until it accepts a new keypress/keyEvent. This is the "time delay" debounce method.

addEventListener(keypadEvent)

Trigger an event if the keypad is used. You can load an example in the Arduino IDE.
[See File -> Examples -> Keypad -> Examples -> EventSerialKeypad] or see the KeypadEvent Example (http://playground.arduino.cc/KeypadTutorial/EventKeypad) code.

For Now

Here's the list of multi-keypress functions and the keylist definition. I will complete their descriptions this weekend.

- Key key[LIST_MAX]
- bool getKeys()
- bool isPressed(char keyChar)
- int findInList(char keyChar)

Example

```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'#','0','*'}
};
byte rowPins[ROWS] = {5, 4, 3, 2}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {8, 7, 6}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
  Serial.begin(9600);
}

void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
  }
}
```

[Get Code] (http://playground.arduino.cc/Code/Keypad?action=sourceblock&num=1)

FAQ

- `How do I use multiple Keypads?`

Keypad is a class. Therefore to use multiple Keypad, you must create an instance for each of them. In the example above, the Keypad instance *keypad*) was bound to the digital pins 2, 3, 4, 5, 6, 7 and 8.

To add a Keypad bound to digital pins 9, 10, 11, 12, 13, 14, 15 and 16, you could create the following instance *keypad2*:

```
const byte ROWS2 = 4; //four rows
const byte COLS2 = 4; //four columns
char keys2[ROWS2][COLS2] = {
  {'.','a','d','1'},
  {'g','j','m','2'},
  {'p','t','w','3'},
  {'*',' ','#','4'}
};
byte rowPins2[ROWS2] = {12, 11, 10, 9}; //connect to the row pinouts of the keypad
byte colPins2[COLS2] = {16, 15, 14, 13}; //connect to the column pinouts of the keypad

Keypad keypad2 = Keypad( makeKeymap(keys2), rowPins2, colPins2, ROWS2, COLS2 );
```

And now it's just a matter of using whatever function is wanted on each keypad:

```
//update instances and possibly fire funcitons
void loop(){
  char key1 = keypad.getKey();
  char key2 = keypad2.getKey();

  if (key1 != NO_KEY || key2 != NO_KEY){
    Serial.print("You pressed: ");
    Serial.print(key1 != NO_KEY ? key1 : "nothing on keypad");
        Serial.print(" and ");
    Serial.print(key2 != NO_KEY ? key2 : "nothing on keypad2");
    Serial.println(".");
  }
}
```

- `How do I use setDebounceTime(unsigned int time)?`

In Arduino follow the main menu from File-> Examples-> Keypad-> Examples-> DynamicKeypad. Once the sketch is open find setup() and there you will see:

```
void setup()  {
  Serial.begin(9600);
  digitalWrite(ledPin, HIGH);            // Turns the LED on.
  keypad.addEventListener(keypadEvent);  // Add an event listener.
  keypad.setHoldTime(500);               // Default is 1000mS
  keypad.setDebounceTime(250);           // Default is 50mS
}
```

This shows that the debounce time will allow one key press every 250 milliseconds. If multiple key presses occur within that time frame (as would happen when a key is bouncing) then those extra presses are simply ignored.

Information about this page

Part of AlphaBeta (http://playground.arduino.cc/Profiles/AlphaBeta) Libraries.

Last Modified:	September 17, 2015, at 08:08 PM
By:	Christopher Andrews

Share

(https://twitter.com/arduino) (http://www.facebook.com/official.arduino)

(https://plus.google.com/+Arduino) (http://www.flickr.com/photos/arduino_cc)

(http://youtube.com/arduinoteam)