The playground is a publicly-editable wiki about Arduino.

edit SideBar

```
I will be maintaining my libraries here:
http://bit.ly/pATDBi

I am the lead developer for libraries
that ship with the Wiring distribution.
As per version 1.0 -
Wiring will support Arduino boards.
You are welcome to check it out!
http://wiring.org.co/download/
```

```
Keypad Library for Arduino
Author:  Mark Stanley, Alexander Brevig
Contact: mstanley@technologist.com, alexanderbrevig@gmail.com
```

Navigation

Current version

2.0 2011-12-29 - Mark Stanley : Added Nick Gammon's changes.

History

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
1.7 2009-06-18 - Alexander Brevig : Added setDebounceTime()
1.6 2009-06-13 - Mark Stanley : getKey() bug fixes
1.5 2009-05-19 - Alexander Brevig : Added setHoldTime()
1.4 2009-05-15 - Alexander Brevig : Added addEventListener()
1.3 2009-05-12 - Alexander Brevig : Added debouncing
1.2 2009-05-09: Changed getKey()
1.1 2009-03-12: Initial Release, NEW Library
1.0 2007: Initial Release

Description

Keypad is a library for using *matrix* style keypads with the Arduino.

This library is based upon the Keypad Tutorial code.

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 libraries.

Version 2.0 has just been posted (28 Dec. 2011) and includes a major rewrite of the state machine for reading keys. Also, attention was given to the speed of operation of this driver because the old keypad driver caused the number of loop()s per second to be around 4000. And that was before ever pressing a key. Version 2.0 brought that number up over 40,000 per second and removed any slowdowns caused when pressing a key.

Since version 1.8 no external diodes or resistors are needed because there is only one column pin driven low at any time and all the other pins are set as inputs with their internal pull-ups enabled. Setting only one pin as an output is also more eco-friendly as it consumes less power.

Download, install and import

Download here: 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()

Initialize all variables
The constructor does this for you.

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 code.

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]

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 key
byte colPins2[COLS2] = {16, 15, 14, 13}; //connect to the column pinouts of the

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 Libraries.

Last Modified: December 31, 2011, at 08:51 PM

By: mstanley

Share |