KBSW180142 SDK-Windows

Windows SDKSDKMFC

-
-
-
- SDK/Slamware

SDKMFC

- Windows
- SDK

SDK

-
-
  -
  -
-
-
-

SDK_Ref_win32.7z

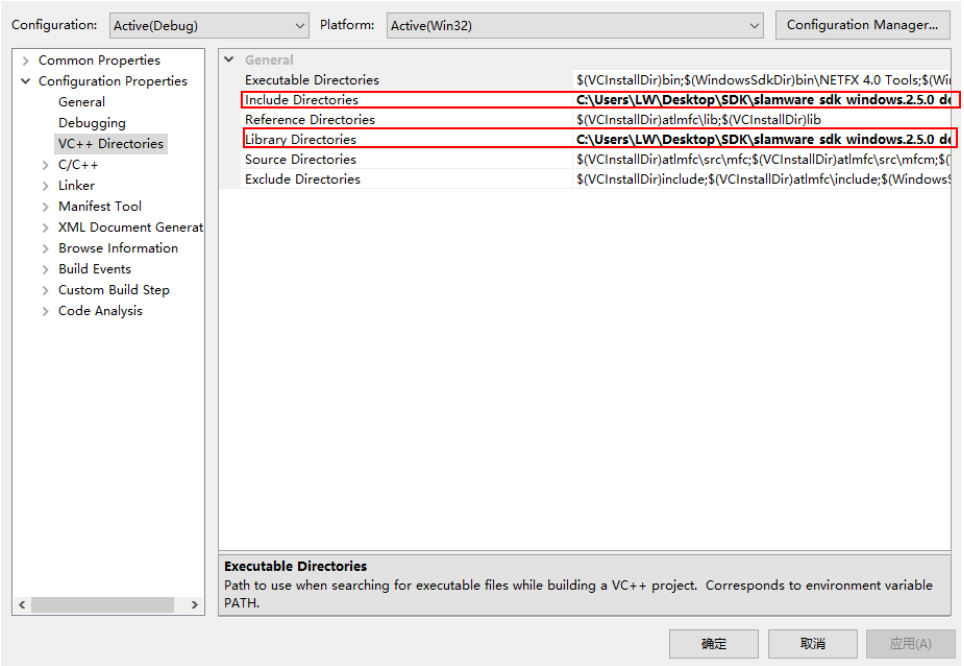
-
  - Visual Studio 2010 SP1
  - Slamware Windows SDK:Slamware Windows SDK
  - RoboStudio():Robostudio installer
  - Sample Code:

    Visual Studio

    Visual Studio 2010SP1.Net FrameworkSP1
-
  - Slamware SDP mini
  - Slamware SDP
  - Zeus/Apollo

    Slamware SDKSlamcoreSDK

1. Slamware SDK includelib

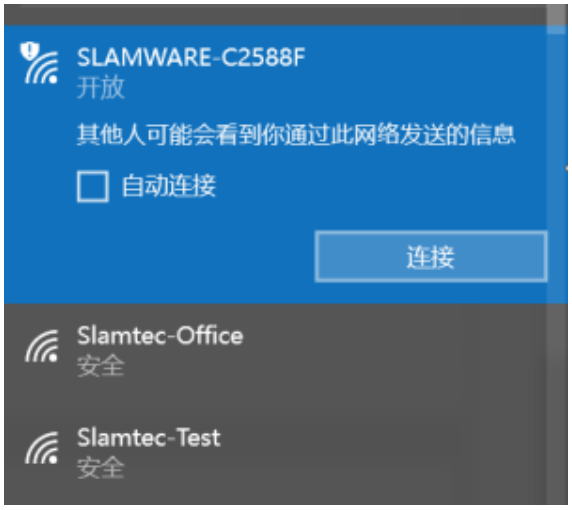


Slamware SDKincludelibVisual Studio

.h

warning

2. wifiSLAMWARE6MAC

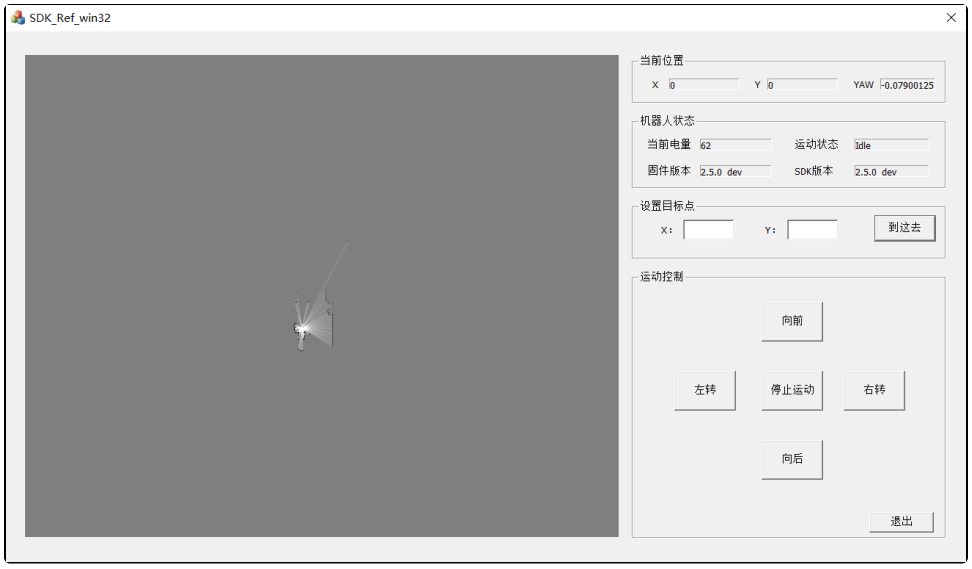


3.

wifiAPIP192.168.11.1StationIPIP

IP192.168.11.1

4.



a.

b. XYXY

c.

d.

1.

a.

moveBy“”

```
void CSDK_Ref_win32Dlg::OnBnClickedBtnForward()
{
        // TODO: Add your control notification handler code here
        try{
                rpos::core::Direction direction(rpos::core::ACTION_DIRECTION::FORWARD);
                moveAction = robot.moveBy(direction);
        }catch(const ConnectionFailException &e)
        {
                MessageBox(_T(""), _T(""), MB_OK);
                CDialogEx::OnCancel();
    } catch(const RequestTimeOutException &e)
        {
                MessageBox(_T(""), _T(""), MB_OK);
    }catch(const std::exception &e)
        {
                MessageBox(_T(""), _T(""), MB_OK);
        }
}
```

b.
moveTo“”

```
void CSDK_Ref_win32Dlg::OnBnClickedBtnMove()
{
        // TODO: Add your control notification handler code here
        double _x = _tstof(m_dest_x);
        double _y = _tstof(m_dest_y);


        try{
                rpos::core::Location loc(_x, _y);
                moveAction = robot.moveTo(loc, false, true);
        }catch(const ConnectionFailException &e)
        {
                MessageBox(_T(""), _T(""), MB_OK);
                CDialogEx::OnCancel();
    }catch(const std::exception &e)
        {
                MessageBox(_T(""), _T(""), MB_OK);
        CDialogEx::OnCancel();
        }
}
```

2.

33100

```
BOOL CSDK_Ref_win32Dlg::OnInitDialog()
{
        CDialogEx::OnInitDialog();
        ......

        // set timer#1 with interval 100 milisecond for status update
        SetTimer(1, 100, NULL);
        // set timer@4 with interval 33 milisecond for map update
        SetTimer(4, 33, NULL);

        ......
}
```

OnTimerOnTimer

```
void CSDK_Ref_win32Dlg::OnTimer(UINT_PTR nIDEvent)
{
        if(nIDEvent == 1)
```

```
	{
		// TODO: Add your message handler code here and/or call default
		UpdateData(TRUE);
		try{
			// update current pose (position + heading angle)
			rpos::core::Pose pose = robot.getPose();
			m_info_x = pose.x();
			m_info_y = pose.y();
			m_info_yaw = pose.yaw();

			// update battery status
			m_info_battery = robot.getBatteryPercentage();

			// update action status
			if(moveAction.isEmpty())
				m_info_action_status = "Idle";
			else
			{
				switch(moveAction.getStatus())
				{
				case ActionStatusWaitingForStart:
					m_info_action_status = "Waiting For Start";
					break;

				case ActionStatusRunning:
					m_info_action_status = "Running";
					break;

				case ActionStatusFinished:
					m_info_action_status = "Finished";
					break;

				case ActionStatusPaused:
					m_info_action_status = "Paused";
					break;

				case ActionStatusStopped:
					m_info_action_status = "Stopped";
					break;

				case ActionStatusError:
					m_info_action_status = "Error";
					break;

				default:
					m_info_action_status = "Error";
					break;
				}
			}
		}catch(const ConnectionFailException &e)
		{
			MessageBox(_T(""), _T(""), MB_OK);
			CDialogEx::OnCancel();
		}catch(const std::exception &e)
		{
			MessageBox(_T(""), _T(""), MB_OK);
			CDialogEx::OnCancel();
		}
		// update the display
		UpdateData(FALSE);
	}
	else
	{
		// update map display
		drawMap();
	}
	CDialogEx::OnTimer(nIDEvent);
}
```

drawMapgetMapDataSlamcoderaw dataBitmap

```
void CSDK_Ref_win32Dlg::drawMap(void)
{
        BITMAPINFO _mapDesc ;
        DWORD* pData ;
        HBITMAP hBitmap ;

        try{
                // get map from slamware
                rpos::core::RectangleF knownArea = robot.getLocationProvider().getKnownArea
(MapTypeBitmap8Bit, location_provider::EXPLORERMAP);
                location_provider::Map map = robot.getMap(MapTypeBitmap8Bit, knownArea, rpos::
features::location_provider::EXPLORERMAP);
                int mapWidth = map.getMapDimension().x();
                int mapHeight = map.getMapDimension().y();

                // initialize the bitmap header structure
                memset(&_mapDesc, 0, sizeof
(_mapDesc));                                                    // clear BITMAPHEADER
                _mapDesc.bmiHeader.biSize = sizeof(_mapDesc);
// size of BITMAPHEADER
                _mapDesc.bmiHeader.biWidth =
mapWidth;                                                      // width by pixels
                _mapDesc.bmiHeader.biHeight =
mapHeight;                                             // height by pixels
                _mapDesc.bmiHeader.biPlanes =
1;                                                              // 1 plane
                _mapDesc.bmiHeader.biBitCount =
32;                                                             // 24bit Color
                _mapDesc.bmiHeader.biCompression =
BI_RGB;                                                // no compression
                _mapDesc.bmiHeader.biSizeImage = mapHeight * mapWidth;                        // size
by pixels

                // create DIB image to fill the map data
                hBitmap = CreateDIBSection (::GetDC(NULL), (BITMAPINFO *) &_mapDesc, DIB_RGB_COLORS,
(void**)&pData, NULL, 0);
                // fill the map data
                for (int posY = 0; posY < mapHeight; ++posY)
                {
                        for (int posX = 0; posX < mapWidth; ++posX)
                        {
                                // get map pixel
                                rpos::system::types::_u8 mapValue_8bit = map.getMapData()[posX + posY
* mapWidth] + 127;
                                // fill the bitmap data
                                pData[posX + posY*mapWidth] = RGB(mapValue_8bit,mapValue_8bit,
mapValue_8bit);
                        }
                }
                // update bitmap map file to picture control component
                m_pic_map.SetBitmap(hBitmap);

        }catch(const ConnectionFailException &e)
        {
                MessageBox(_T(""), _T(""), MB_OK);
                CDialogEx::OnCancel();
    }catch(const std::exception &e)
        {
                MessageBox(_T(""), _T(""), MB_OK);
        CDialogEx::OnCancel();
        }
}
```

33drawMap33ms

-Slamware